A. Background for Diffusion Model

A diffusion model [19,57] is defined by a forward process that gradually corrupts data $\boldsymbol{\tau}^0 \sim q(\boldsymbol{\tau}^0)$ over T timesteps

$$\begin{aligned} q(\boldsymbol{\tau}^{1:T}|\boldsymbol{\tau}^0) &= \prod_{t=1}^{T} q(\boldsymbol{\tau}^t|\boldsymbol{\tau}^{t-1}) \\ q(\boldsymbol{\tau}^t|\boldsymbol{\tau}^{t-1}) &= \mathcal{N}(\boldsymbol{\tau}^t; \sqrt{1-\beta^t}\boldsymbol{\tau}^{t-1}, \beta\mathbf{I}) \end{aligned} \tag{A1}$$

and a reverse process $p_\theta(\boldsymbol{\tau}^0) = \int p_\theta(\boldsymbol{\tau}^{0:T}) d\boldsymbol{\tau}^{1:T}$ where

$$\begin{aligned} p_\theta(\boldsymbol{\tau}^{0:T}) &= p(\boldsymbol{\tau}^T) \sum_{t=1}^{T} p_\theta(\boldsymbol{\tau}^{t-1}|\boldsymbol{\tau}^t) \\ p_\theta(\boldsymbol{\tau}^{t-1}|\boldsymbol{\tau}^t) &= \mathcal{N}(\boldsymbol{\tau}^{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{\tau}^t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{\tau}^t, t)). \end{aligned} \tag{A2}$$

The forward process hyperparameters β^t are set so that $\boldsymbol{\tau}^T$ is approximately distributed according to a standard normal distribution, so $\boldsymbol{\tau}^T$ is set to a standard normal prior as well. The reverse process is trained to match the joint distribution of the forward process by optimizing the evidence lower bound (ELBO) [19,57]. As suggested by the literature [19, 41], we can use the reverse process parametrizations as:

$$\begin{aligned} \boldsymbol{\mu}_\theta(\boldsymbol{\tau}^t, t) &= \frac{1}{\sqrt{\alpha^t}}(\boldsymbol{\tau}^t - \frac{\beta^t}{\sqrt{1-\hat{\alpha}^t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}^t, t)) \\ \boldsymbol{\Sigma}_\theta^{ii}(\boldsymbol{\tau}^t, t) &= \exp(\log\hat{\beta}^t + (\log\beta^t - \log\hat{\beta}^t)v_\theta^i(\boldsymbol{\tau}^t, t)) \end{aligned} \tag{A3}$$

where $\alpha^t = 1-\beta^t$, $\hat{\alpha}^t = \sum_{s=1}^{t}\alpha^s$, and $\hat{\beta}^t = \frac{1-\hat{\alpha}^{t-1}}{1-\hat{\alpha}^t}\beta^t$.

We can optimize modified loss instead of the ELBO to improve the sample quality, depending on whether we learn $\boldsymbol{\Sigma}$ or treat it as a fixed hyper-parameter. For the non-learned case, we use the simplified loss:

$$\begin{aligned} \mathcal{L}_{simple}(\theta) &= \mathbb{E}_{t,\boldsymbol{\epsilon},\boldsymbol{\tau}^0}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\hat{\alpha}^t}\boldsymbol{\tau}^0 + \sqrt{1-\hat{\alpha}^t}\boldsymbol{\epsilon}, t)\|^2\right] \\ &= \mathbb{E}_{t,\boldsymbol{\epsilon},\boldsymbol{\tau}^0}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\boldsymbol{\tau}^t, t)\|^2\right] \end{aligned} \tag{A4}$$

It is a weighted form of the ELBO that resembles denoising score matching over multiple noise scale [19,59].

Conditional Diffusion Model The conditional diffusion model aims to learn a conditional distribution $p_\theta(\boldsymbol{\tau}^0|\mathbf{c})$. We modify the diffusion model to include the condition $\mathbf{c}$ as input to the inverse process:

$$\begin{aligned} p_\theta(\boldsymbol{\tau}^{0:T}|\mathbf{c}) &= p(\boldsymbol{\tau}^T) \prod_{t=1}^{T} p_\theta(\boldsymbol{\tau}^{t-1}|\boldsymbol{\tau}^t, \mathbf{c}) \\ p_\theta(\boldsymbol{\tau}^{t-1}|\boldsymbol{\tau}^t, \mathbf{c}) &= \mathcal{N}(\boldsymbol{\tau}^{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{\tau}^t, t, \mathbf{c}), \boldsymbol{\Sigma}_\theta(\boldsymbol{\tau}^t, t, \mathbf{c})) \end{aligned} \tag{A5}$$

B. Model Architectures

For human pose/motion generation tasks in 3D scenes and path planning for 3D scene navigation, we use the same scene encoder, *i.e.*, the PointTransformer [91] adopted from the original architecture. We pre-train the scene encoder with indoor scene semantic segmentation task on ScanNet dataset and freeze it while training SceneDiffuser. The outputs of the scene encoder are used as the key and value of the cross-attention module.

For processing the trajectory, we employ an FC layer and positional embedding to obtain the high-dimensional feature of the trajectory. We then fuse the trajectory feature with denoising timestep embedding with a ResBlock. After that, we feed the fused feature vectors to a self-attention module and use them as the query of the cross-attention module. Finally, the computed vector is fed into a feedforward layer to estimate the noise $\boldsymbol{\epsilon}$.

For the task of dexterous grasp generation for 3D objects, we use PointNet [45] as the 3D object encoder. Before the cross-attention module, the outputs of PointNet are reshaped to $(N_{\text{token}}, N_{\text{feat}})$, where N_{token} refers to the number of tokens and N_{feat} refers to the dimensions of the feature.

For the task of motion planning for robot arms, we adopt PointTransformer [91] as the scene encoder, which is jointly trained from scratch with SceneDiffuser.

C. Objective Design

For **human pose and motion generation** in 3D scenes, we encourage contact and non-collision between the generated human body and scene meshes. Following Zhang *et al*. [90], we design optimization objective $\varphi_o(\boldsymbol{\tau}^t|\mathcal{S}) = \alpha_1\varphi_o^{\text{collision}} + \alpha_2\varphi_o^{\text{contact}}$ for pose generation and $\varphi_o(\boldsymbol{\tau}^t|\mathcal{S}) = \alpha_1\varphi_o^{\text{collision}} + \alpha_2\varphi_o^{\text{contact}} + \alpha_3\varphi_o^{\text{smoothness}}$ for motion generation. α is the balancing weight. $\varphi_o^{\text{collision}}$ minimizes the negative signed-distance values of the body mesh vertices given the negative signed distance field (SDF) of the 3D scene $\Phi_s^-(\cdot)$, which is formulated as

$$\varphi_o^{\text{collision}} = -\mathbb{E}\left[|\Phi_s^-(\mathcal{M}^t)|\right], \tag{A6}$$

where $\mathcal{M}^t$ is the SMPL-X body mesh at denoising step t. $\varphi_o^{\text{contact}}$ minimize the distance between contact body parts of the generated body mesh and the scene mesh, which is formulated as

$$\varphi_o^{\text{contact}} = -\sum_{v_c \in C(\mathcal{M}^t)} \min_{v_s \in \mathcal{S}} |v_c - v_s|, \tag{A7}$$

where $C(\cdot)$ is the operation of selecting contact part vertices from the SMPL-X body mesh according to the annotation in Hassan *et al*. [15]. We design the smoothness objective to smooth the motion over time by minimizing the velocity difference of consecutive frames, which is formulated as

$$\varphi_o^{\text{smoothness}} = -\sum_{v \in \mathcal{M}^t} \sum_{i=1}^{L-2} \|v^{i+2} - 2v^{i+1} + v^i\|^2, \tag{A8}$$

where L is the length of the motion sequence. We empirically set $\alpha_1 = 1.0$, $\alpha_2 = 0.02$, and $\alpha_3 = 0.001$.

We punish the collision between the robotic hand mesh and object mesh for **dexterous grasp generation**. We design optimization $\varphi_o(\boldsymbol{\tau}^t|\mathcal{S}) = \varphi_o^{\text{collision}}$. $\varphi_o^{\text{collision}}$ is similar to

Eq. (A6), where 3D scene is represented as an object and $\mathcal{M}^t$ as the robotic hand mesh at denoising step t.

For **path planning** for the 3D scene navigation task, we design an optimization objective φ_o and φ_p for generating collision-free paths toward the goals. The collision-free objective maximizes the distance between the robot and the scene vertices in the robot cylinder, formulated as

$$\varphi_o = -\sum_{i=1}^{L}\sum_{v_s\in\mathcal{S}} \text{ReLU}(r - \text{dist}(v_s, \tau_i^t)), \tag{A9}$$

where $\text{ReLU}(x) = \max(0, x)$, r is the radius of the robot cylinder, and $\text{dist}(\cdot)$ compute the Euler distance between scene vertices and robot position on the 2D plane. The planning objective φ_p encourages the generated paths toward the target position. In our work, we formulate it as

$$\varphi_p = \sum_{i=1}^{L} \exp\left(\frac{1}{\|\mathcal{G} - \tau_i^t\|_1}\right). \tag{A10}$$

We design the planning objective φ_p similar to Eq. (A10) for robot arm **motion planning**. The objective is defined as

$$\varphi_p = \sum_{i=1}^{L} \exp\left(\frac{1}{\sum_{j=1}^{N}\|\mathcal{G}_j - \tau_{ij}^t\|_1}\right). \tag{A11}$$

where N denotes to the number of revolute joints and j refers to j-th revolute joint.

D. Implementation Details

D.1. Human Pose Generation in 3D Scenes

Following prior work [90], we represent the human body with the SMPL-X model. We denote the parameters of SMPL-X in our setting as $x_h \coloneqq (t, R, \beta, \theta_b)^T \in \mathbb{R}^{79}$, where t is the global translation in meters, R is the global orientation represented in axis-angle, $\beta \in \mathbb{R}^{10}$ is the body shape feature, and $\theta_b \in \mathbb{R}^{63}$ is the axis-angle representation of 21 body joints. SMPL-X can map these low-dimensional parameters into a watertight mesh with a fixed topology, enabling physical collision and contact modeling. Unlike [90] using scene depth and semantics, we directly represent the scene with a point cloud $\mathcal{S} \in \mathbb{R}^{32768\times 3}$, which provides raw information about the 3D scene.

We randomly sample 1000 examples in each test scene for quantitative evaluation to compute the diversity and physics metrics. Specifically, we separately calculate the Average Pairwise Distance (APD) and standard deviation (std) for global translation $t \in \mathbb{R}^3$, the rest of local SMPL-X parameters $(R, \beta, \theta_b)^T \in \mathbb{R}^{76}$, and the marker-based representation [89] of generated bodies without global translation. We also report the non-collision score of the generated human bodies by calculating the proportion of the scene vertices with positive SDF to the human body and the contact score by checking whether the body contacts with the scene within a pre-defined distance threshold, *i.e.*, 0.02m.

To train SceneDiffuser, we use Adam [27] optimizer with 0.0001 as the learning rate. We use 4 NVIDIA A100 GPUs to train 100 epochs with a batch size of 128. The number of diffusion steps T in this task is set as 100. For optimization guidance sampling, we empirically set scale coefficient $\lambda = 2.5$.

D.2. Human Motion Generation in 3D Scenes

For the two different settings (with and without start position) of human motion generation in 3D scenes, we represent the single-frame human body of the motion sequence as the same as the pose generation. To collect training data, we clip the motion sequences in the PROX dataset into motion segments with a fixed duration, *i.e.*, 60 frames. We use the same evaluation metrics as pose generation and report the average values over motion sequence as the motion generation performance measure. In this task, the optimizer is Adam, and the learning rate is 0.0001. We use 4 NVIDIA A100 GPUs to train 300 epochs with 200 diffusion steps and a batch size of 128. For optimization guidance sampling, we empirically set scale coefficient $\lambda = 2.5$.

D.3. Dexterous Grasp Generation for 3D Objetcs

We use Shadowhand as our dexterous robotic hand. We denote qpos as $q \coloneqq (t, R, \theta) \in \mathbb{R}^{33}$, where $t \in \mathbb{R}^3$ and $R \in \mathbb{R}^6$ represent the global translation and orientation respectively, $\theta \in \mathbb{R}^{24}$ describes the rotation angles of the revolute joints. We split the MultiDex [31] into 48 seen objects and 10 unseen objects for training and testing.

For each grasp, we apply 0.5ms^{-2} acceleration to the object along $\pm xyz$ directions, and the grasping is successful if the movements of the object are all within 2cm. For the diversity, we first capture the mean μ_i and the standard deviation σ_i of i revolute joint in the training data grasping pose. We define the mean pose as $\mu_q \coloneqq (\mu_1, \mu_2, ..., \mu_{24}) \in \mathbb{R}^{24}$ and the standard deviation pose as $\sigma_q \coloneqq (\sigma_1, \sigma_2, ..., \sigma_{24}) \in \mathbb{R}^{24}$. We report the success rate of generated poses that lie at the k standard deviation level, which means these poses q satisfy the constraint as $\mu_q - k\sigma \leqslant q \leqslant \mu_q + k\sigma$. For the depth collision computation, we sample the surface points $\mathcal{H} \in \mathbb{R}^{3200\times 3}$ on the ShadowHand related to the pose q and the surface points with normal $\mathcal{O} \in \mathbb{R}^{4096\times 6}$ on the object. We compute the collision for ShadowHand surface to the object and report the depth collision among $\mathcal{H}$ to show the quality of generated poses.

To train SceneDiffuser on this task, we use Adam optimizer, set the learning rate as 0.0001, and use 1 NVIDIA A100 GPU to train 2100 epochs with 64 batch size. For optimization guidance sampling, we empirically set scale coefficient $\lambda = 1.0$.

D.4. Path Planning for 3D Scene Navigation

In this task, we consider 3D navigation in realistic scenes, where the goal is to plan plausible trajectories for a physical robot from the given start position $\hat{\mathbf{s}}_0$ to the given target position $\mathcal{G}$ in a furnished 3D indoor scene $\mathcal{S}$. We represent the hallucinated physical robot as a cylinder to simulate physically plausible trajectories which are collision-free in the 3D scene. The robot can move in all directions within a distance in each step without height change. We set the maximum moving distance as 0.08m, the robot radius as 0.08m, and the robot height as infinite, which means the robot can only move on the floor that is not occupied.

To construct room-level realistic scenarios for path planning, we select 61 indoor scenes from ScanNet [9], as shown in Fig. A1. We annotate these scenes with spatially dense and physically plausible navigation graphs and collect about 6.3k trajectories by searching the shortest paths between the randomly selected start and target graph nodes. As the distance between nodes may be too long for a robot to move in one step, we refined the trajectories according to the maximum moving distance. These trajectories have an average step of 60.0, a minimal step of 32, and a maximum step of 120. We use 4.7k trajectories in 46 scenes as the training data and the rest 1.6k trajectories in 15 unseen scenes for evaluation. We set the maximum number of planning steps as 150.

During training, we set the fixed trajectory horizon as 32. We use 4 NVIDIA A100 GPUs to train 50 epochs with 100 diffusion steps and a batch size of 128. The optimizer is Adam, and the learning rate is 0.0001. During inference, we empirically set the scale coefficient of optimization guidance as 1.0 and the scale coefficient of planning guidance as 0.2.

D.5. Motion Planning for Robot Arm

We use the Franka Emika with seven revolute joints as the robot arm and randomly generate cluttered tabletop scenes with primitives following specific heuristics. For each scene, we position the robot arm at the center of the table and use moveit2 motion planner [50] to synthesize trajectories constrained by a pair of start and goal poses of the end effector. We collected 19,800 collision-free trajectories over 200 clustered scenes.

During inference, we execute the planned motions of SceneDiffuser in IsaacGym [37]. We consider the planning is successful if our robot arm reaches the goal pose by a certain L_2 norm distance (e.g., 0.2) in the space of revolute joints. Note that the simulation can not run infinitely; therefore, we set a limited number of simulation steps (e.g., 300). For the efficiency evaluation, we capture the average number of simulation steps.

To train SceneDiffuser on this task, we use Adam optimizer, set the learning rate as 0.0001, and use 4 NVIDIA A100 GPUs to train 200 epochs with a batch size of 128. We empirically set the scale coefficient of planning guidance as 0.2 during inference.

D.6. Scaling Factor for the Guidance

Like Ho *et al*. [19], we notice that the parameter Σ in Eq. (10) decreases as the denoising step t decreases, which gradually weakens the guidance during the denoising process. Instead of using a constant as the scaling factor, we empirically schedule the scaling factor by dividing it by Σ. It reformulates Eq. (10) as

$$p(\boldsymbol{\tau}^{t-1}|\boldsymbol{\tau}^t, \mathcal{S}, \mathcal{G}) \approx \mathcal{N}(\boldsymbol{\tau}^{t-1}; \boldsymbol{\mu} + \lambda \mathbf{g}, \boldsymbol{\Sigma})$$

E. Additional Ablative Experiments

We ablate different model architectures, including the scene encoder and noise prediction module in SceneDiffuser, diffusion steps and scale coefficient in the optimizer of dexterous grasp generation task, and fixed frames and planning objectives of path planning for 3D scene navigation task.

E.1. Model Architecture

As shown in Tab. A1, we study how different scene model influences the dexterous grasp generation results. We use PointNet [45] and PointNet++ [46] as different scene models to extract the object feature. For more diversity evaluation, we capture the mean standard deviation among all revolute joints of the robotic hand qpos. We find that the global feature extracting from PointNet makes it easier for the model to learn a mean pose to obtain a higher grasping success rate. In contrast, the local feature extracted from PointNet++ makes the generated grasp pose more diverse.

Table A1. **Ablation on different scene encoder.**

scene encoder	**succ. rate** (%)↑			**div.** (rad.)↑	**coll.** (mm)↓
	σ	2σ	all		
PointNet (w/o opt.)	70.65	71.25	71.25	0.0718	17.34
PointNet (w/ opt.)	71.27	70.32	69.84	0.0838	14.61
PointNet++ (w/o opt.)	56.47	66.29	66.25	0.1568	18.53
PointNet++ (w/ opt.)	64.33	60.51	59.53	0.1670	14.37

As shown in Tab. A2, we ablate the module for noise prediction. We compare the design of cross-attention and self-attention for processing the condition and input. Cross-attention indicates learning query from the input $\boldsymbol{\tau}_t$ and learning key and value from the scene condition $\mathcal{S}$. Self-attention indicates concatenating $\boldsymbol{\tau}_t$ and scene features $\mathcal{S}$ and learning with self-attention. Through our experiments, we find that with self-attention, the model learns better to capture the joint distribution of input and condition. This leads to a slightly lower diversity but better generation quality and success rate.

Table A4. **Ablation on different inpainting horizons and scale coefficients of the planning guidance.**

fixed frames	planner scale	succ. rate(%)↑	planning steps↓
1	0.2	31.25	135.14
7	0.2	65.50	104.30
15	0.2	73.75	90.38
23	0.2	73.25	87.49
31	0.1	53.50	106.23
	0.2	62.37	97.02
	0.3	56.81	101.54
	0.4	50.94	105.11

Table A2. **Ablation on different model architecture.**

epsilon model	succ. rate (%)↑			div. (rad.)↑	coll. (mm)↓
	σ	2σ	all		
CrossAttn. (w/o opt.)	70.65	71.25	71.25	0.0718	17.34
CrossAttn. (w/ opt.)	71.27	70.32	69.84	0.0838	14.61
SelfAttn. (w/o opt.)	74.27	75.94	75.94	0.0535	16.49
SelfAttn. (w/ opt.)	72.01	71.56	71.09	0.0605	13.94

E.2. Diffusion Steps

We study different diffusion steps T in Tab. A3, using PointNet++ as the scene encoder with cross-attention design. We report the success rate, diversity, and depth collision of sampling results in the test set under different diffusion steps, ranging from 30 to 1000. T balance the diversity and success rate in dexterous grasp generation, where $T = 30$ leads to the best diversity of generated grasp pose and $T = 1000$ leads to the best *all* success rate.

Table A3. **Ablation on diffusion steps and scale coefficient.**

time steps	optimizer scale	succ. rate (%)↑				div. (rad.)↑	coll. (mm)↓
		σ	2σ	3σ	all		
30	w/o	0.00	60.01	50.94	48.13	0.3418	21.19
30	0.1	0.00	58.72	54.90	51.09	0.3415	19.96
30	0.5	0.00	64.24	51.63	47.81	0.3397	17.41
30	1.0	0.00	60.41	48.76	43.59	0.3393	16.05
100	w/o	0.00	66.62	60.12	58.91	0.2865	19.07
100	0.1	0.00	66.54	60.60	59.53	0.2836	17.55
100	0.5	0.00	61.23	56.71	53.75	0.2898	14.63
100	1.0	0.00	56.79	53.13	48.91	0.2920	14.53
500	w/o	75.00	67.50	67.34	67.34	0.1753	19.29
500	0.1	68.56	65.19	65.00	65.00	0.1733	17.68
500	0.5	62.83	60.25	58.94	58.75	0.1814	15.12
500	1.0	62.21	57.76	55.17	54.37	0.1872	14.36
1000	w/o	56.47	66.29	66.26	66.25	0.1568	18.53
1000	0.1	73.24	71.43	71.04	71.09	0.1572	16.88
1000	0.5	70.18	65.99	65.55	65.62	0.1611	14.37
1000	1.0	64.33	60.51	59.61	59.53	0.1670	14.37

E.3. Scale Coefficient

Among different time steps T, we ablate scale coefficient λ of the optimization guidance in dexterous grasp generation in Tab. A3, ranging from 0.0 (denoted as w/o in the table) to 1.0. Through extensive experiments, we observe that, in general, the α trades off the depth collision and grasp success rate. A larger α value leads to fewer collisions and draws the grasp pose away from the object simultaneously, which losses the grasp stability and lowers the success rate.

We also ablate the scale coefficient of the planner in path planning for 3D scene navigation, as shown in Tab. A4. Too small or too large scale coefficients both lead to a performance drop. A small value cannot provide sufficient guidance, whereas a large value diminishes trajectory diversity with strong guidance, preventing it from escaping obstacles and dead-ends.

E.4. Fixed Frames for Planning

Since we formulate the planning algorithm as inpainting, we also ablate the number of the fixed frame in it. In path planning for 3D scene navigation, we train the SceneDiffuser with a trajectory length of 32. Therefore, we compare the settings of fixing the first 1, 7, 15, 23, and 31 frames for inpainting during the denoising process. The results in Tab. A4 show that the model achieves the best performance while fixing the first 15 frames.

E.5. Planning Objectives

To explore the influence of different planning objectives, we design the following four planning objectives and compare them with Eq. (A10).

- We compute the L1 distance between the last frame of the denoised trajectory and the target position, *i.e.*,

$$\varphi_p = -\|\mathcal{G} - \tau_L^t\|_1. \tag{A12}$$

- We summarize the L1 distance between all frames of the denoised trajectory and the target position, *i.e.*,

$$\varphi_p = -\sum_{i=1}^{L} \|\mathcal{G} - \tau_i^t\|_1. \tag{A13}$$

- Similar to Eq. (A10), we only consider the last frame of the denoised trajectory, *i.e.*,

$$\varphi_p = \exp\left(\frac{1}{\|\mathcal{G} - \tau_L^t\|_1}\right). \tag{A14}$$

- We compute the L1 distance between the target position and the frame closest to the target, *i.e.*,

$$\varphi_p = -\min_i \|\mathcal{G} - \tau_i^t\|_1. \tag{A15}$$

The planning results in Tab. A5 indicate that encouraging all frames of the denoised trajectory to reach the target position surpasses considering only one frame. Besides, directly using $L1$ distance tends to perform better than additionally applying the exponential function.

Table A5. **Ablation on different planning objectives.**

objective	succ. rate(%)↑	planning steps↓
$\varphi_p = -\|\mathcal{G} - \tau_L^t\|_1$	57.06	116.22
$\varphi_p = -\sum_{i=1}^{L} \|\mathcal{G} - \tau_i^t\|_1$	75.69	88.02
$\varphi_p = \exp\left(\frac{1}{\|\mathcal{G} - \tau_L^t\|_1}\right)$	34.31	131.74
$\varphi_p = \sum_{i=1}^{L} \exp\left(\frac{1}{\|\mathcal{G} - \tau_i^t\|_1}\right)$	73.75	90.38
$\varphi_p = -\min_i \|\mathcal{G} - \tau_i^t\|_1$	56.00	109.02

F. Trainable Optimization and Planning

As shown in Alg. 1, we can optionally train the optimization and planning process with observed trajectories. We optimize the trainable scaling factor λ of the optimization guidance in pose generation and path planning tasks to verify its efficacy. Specifically, we use a small MLP model to map the timestep embedding of each step into a scalar, *i.e.*, the scaling factor. We only optimize the MLP while fixing the pre-trained diffusion model during training. We plot the learned scaling factor varying with the denoising step from 100 to 1, as shown in Fig. A2. We observe that the scaling factor of the denoising process at the beginning is much smaller than at the end. We speculate that the target signal at the beginning of the denoising process is mostly noise, such that a large scaling factor cannot optimize it properly. The scaling factor decrease in the last several steps may be because this can alleviate excessive guidance and balance the guidance from other modules, such as the planner.

G. More Qualitative Results

Pose Generation in 3D Scenes We show more qualitative results in Fig. A3.

Motion Generation in 3D Scenes In other scenes, we provide more sampled human motions from the same start pose, as shown in Fig. A4. Please refer to the supplemental demo video for better visualization with rendered animations.

Path Planning for 3D Scene Navigation Fig. A5 shows some qualitative results of path planning for 3D scene navigation.

Dexterous Grasp Generation for 3D Objects We show more qualitative results in Fig. A6. Note that the objects are unseen during training time.

Motion Planning for Robot Arm We render the planning results into animations for visualization. Please refer to the supplemental demo video for the qualitative results.

H. Limitation

The primary limitation of the SceneDiffuser is its slow training and test speed compared to previous scene-conditioned generative models, a common issue of diffusion-based methods. We also observe that the optimization and planning are highly dependent on the objective designs, which require efforts on hyper-parameter tuning.

I. Future Work

A promising future direction is extending SceneDiffuser to richer 3D representations, including RGB-D images, semantic images, bird-eye view (BEV) images, videos, 3D meshes, and neural radiance field (NeRF) [38]. Such flexible conditions consume a tremendous amount of 3D training data, which is also a significant challenge. We also hope to extend the SceneDiffuser to outdoor scenes, *e.g.*, the autonomous driving scenarios [3]. Moreover, the SceneDiffuser can be combined with recent large language models (LLMs) [2] for automatic generation and planning with natural language instructions in 3D scenes, which is promising for the vision and robotics community. Finally, SceneDiffuser can serve as the tool for analyzing the behaviors of humans and agents if we can properly learn the planning objective, which naturally encodes the values and preferences that underlie the trajectories.

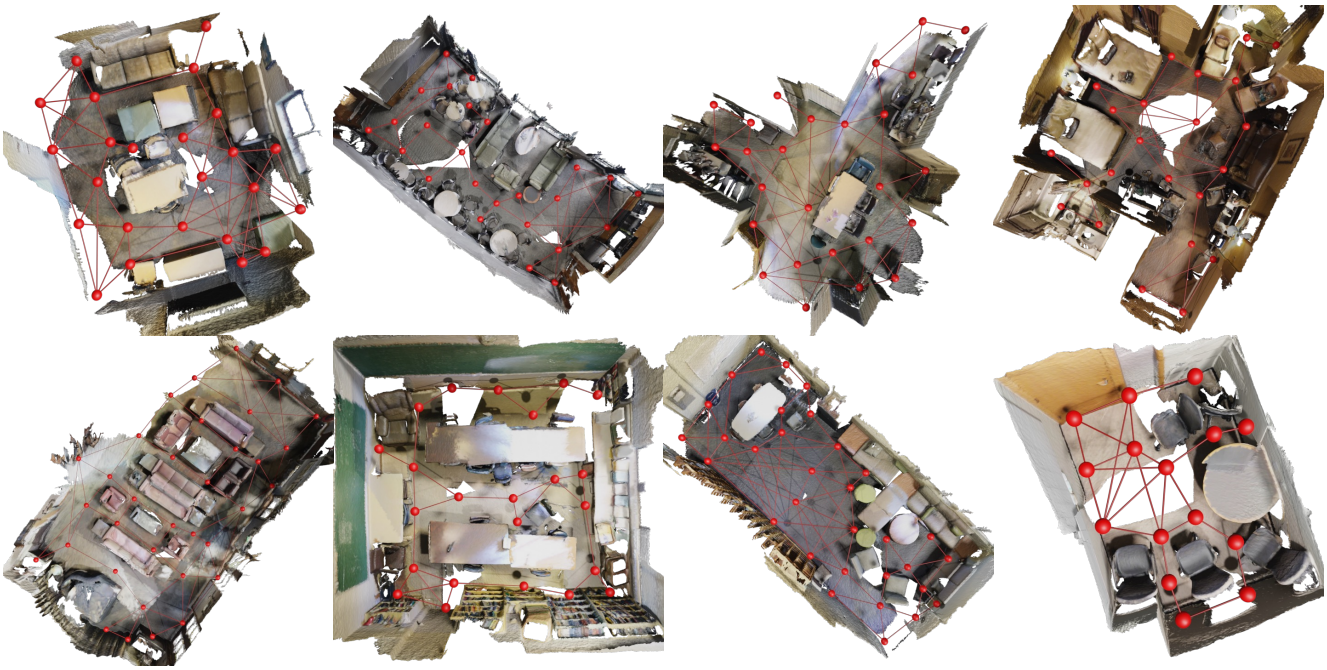
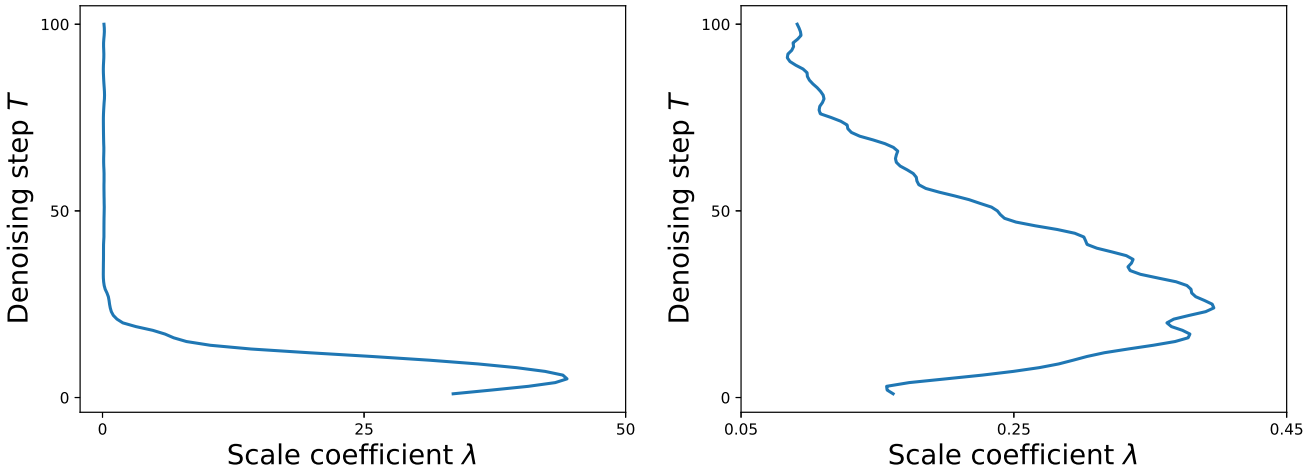
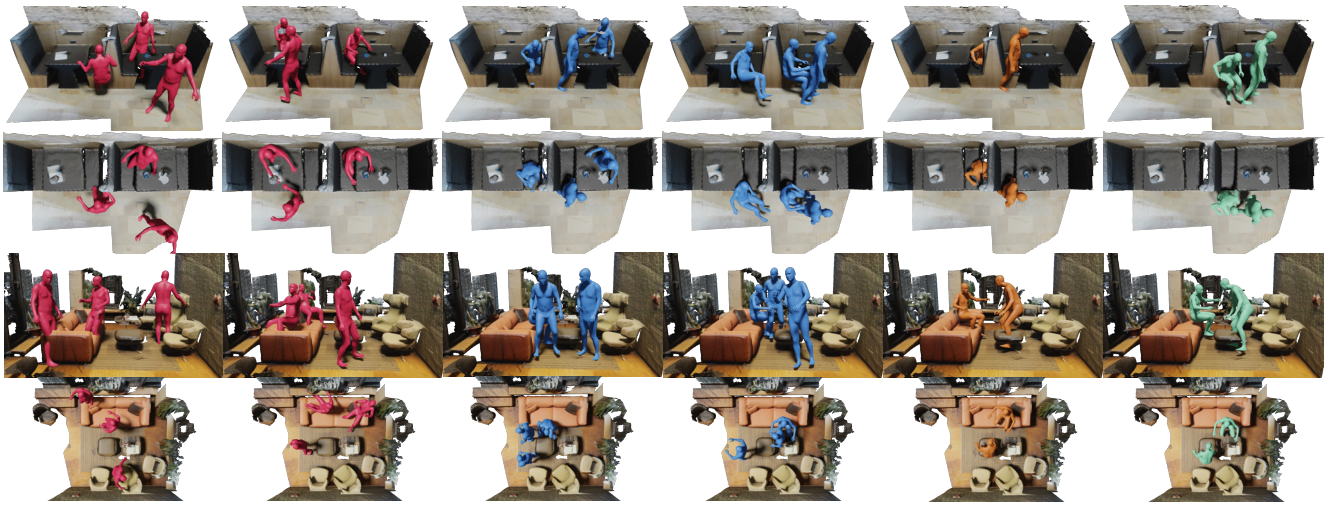


Figure A1. **Scenes and corresponding navigation graphs for path planning.** The selected scenes have various regions, diverse room types, and sufficient layout complexity.



(a) pose generation (b) path planning

Figure A2. **Trainable scaling factor varying with the denoising step.**



(a) cVAE (b) SceneDiffuser (c) Without Opt. (d) With Opt.

Figure A3. **More qualitative results of pose generation in 3D scenes.**

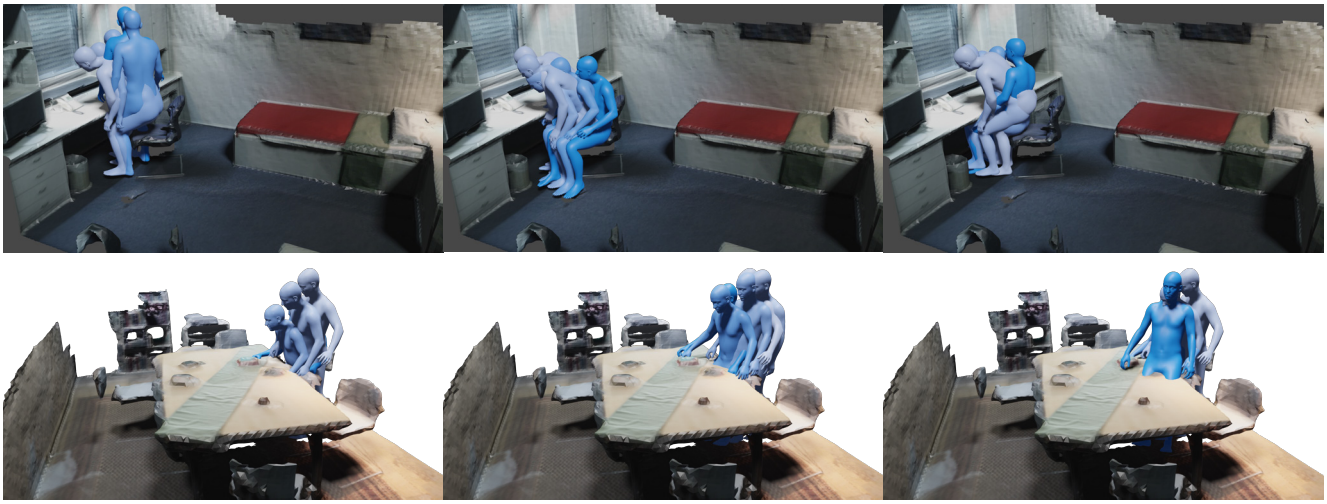


Figure A4. **More qualitative results of motion generation in 3D scenes.**

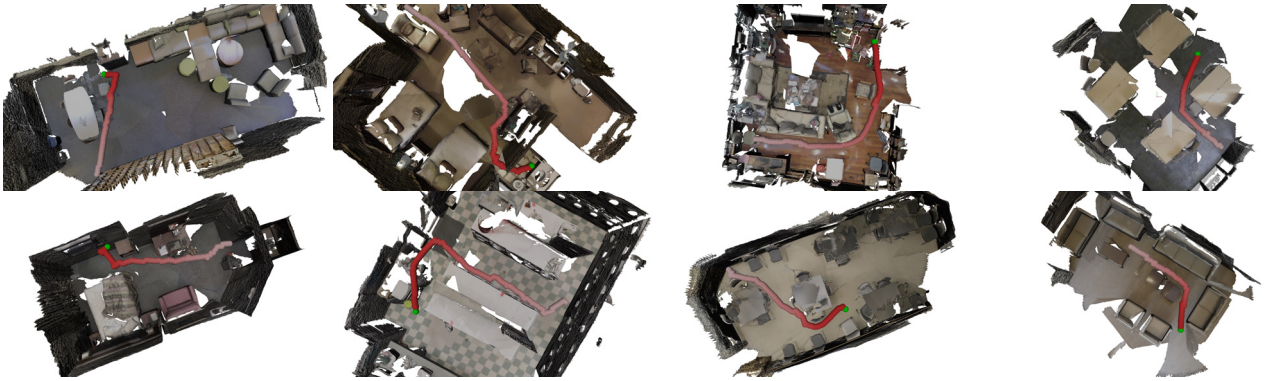


Figure A5. **Qualitative results of path planning for 3D scene navigation.** The red balls represent the planning result, starting with the lightest red ball and ending with the darkest red ball. The green ball indicates the target position.

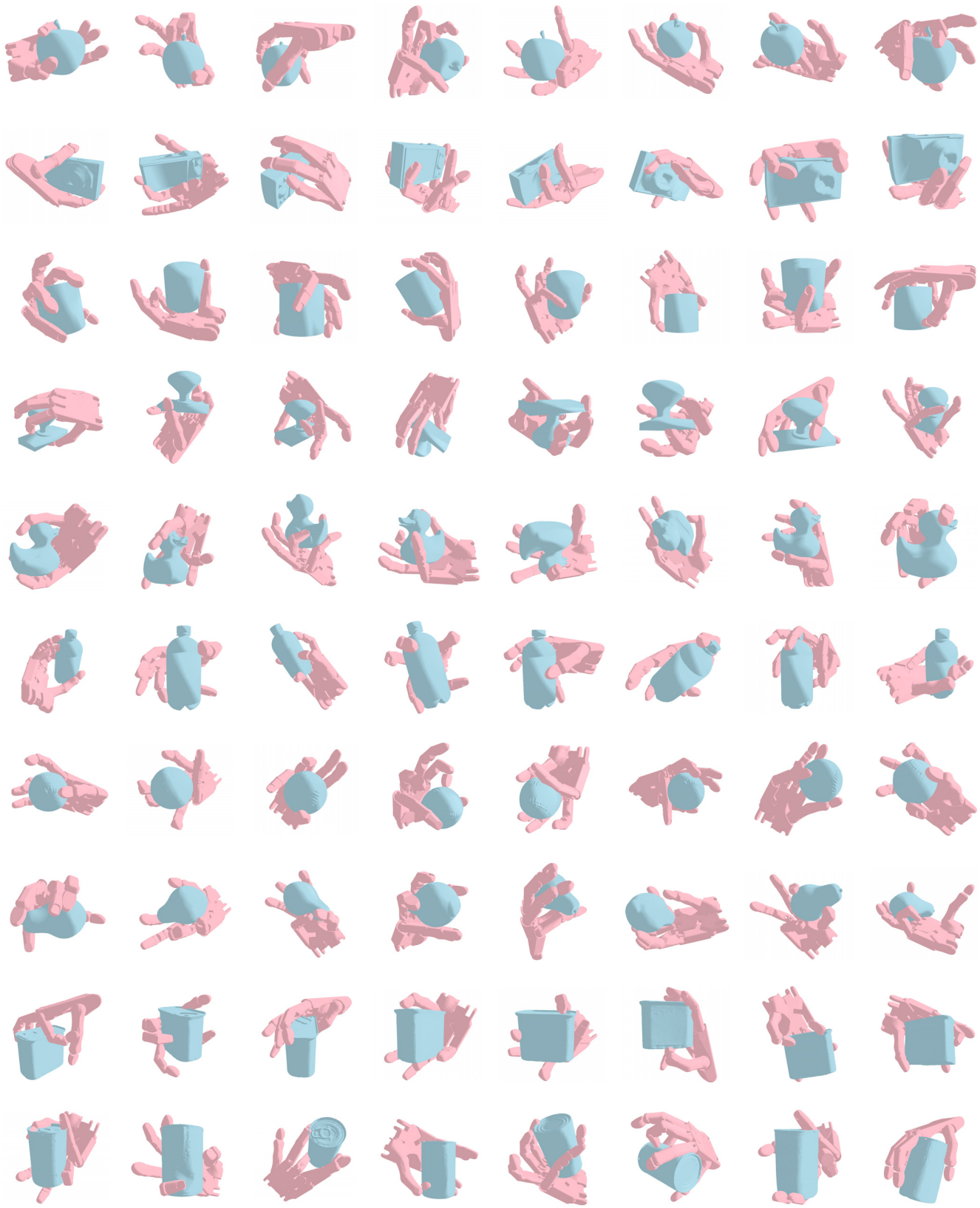


Figure A6. **Additional qualitative results of dexterous grasp pose generation for 3D object.**