A EXPERIMENT DETAILS

To enhance the reproducibility of NeRFuser, we summarize the experiment details such as the configurations and parameters used in Table 6.

Table 6: Hper-parameters. We list all hyper-parameters for reproducing NeRFuser results below.

Parameter	**Value**
training iteration	50K
Learning rate Scheduler	False
Optimizer	Adam
learning rate	0.0005
Weight Decay	1×10^{-6}
Translation loss coefficient (λ_{trans})	300
Rotation loss coefficient (λ_{trans})	1
Gripper openness loss coefficient (λ_{trans})	1
Collision loss coefficient (λ_{trans})	1
Diffusion loss coefficient (λ_{diff})	5
denoising steps (training)	100
ray batch size b_{ray}	512
embedding loss coefficient λ_{feat}	0.01
Batch size	32
GPU	RTX 3090

B SIMULATED TASK DESCRIPTIONS

We have chosen ten language-conditioned tasks from RLBench (James et al., 2020). An overview of these tasks is presented in Table 7. Our variations include randomly sampled colors, sizes, placements, and object categories. The color set includes twenty instances: red, maroon, lime, green, blue, navy, yellow, cyan, magenta, silver, gray, orange, olive, purple, teal, azure, violet, rose, black, and white. The size set comprises two types: short and tall. The placements and object categories are specific to each task. The average keyframes numbers are varied from 2 to 15, representing different horizon length.

Table 7: **Language-conditioned tasks in RLBench (James et al., 2020).**

Task	Variation Type	# of Variations	Avg. Keyframs	Language Template
`turn tap`	placement	2	2.0	"turn — tap"
`drag stick`	color	20	6.0	"use the stick to drag the cube onto the — — target"
`open fridge`	placement	1	4.4	"open the fridge door"
`put in drawer`	placement	3	15.0	"put the item in the — drawer"
`sweep to dustpan`	size	2	4.6	"sweep dirt to the — dustpan"
`meat off grill`	category	2	5.0	"take the — off the grill"
`phone on base`	placement	1	6.4	"put the phone on the base"
`place wine`	placement	3	6.2	"stack the wine bottle to the — of the rack"
`slide block`	color	4	4.7	"slide the block to — target"
`put in safe`	placement	3	6.1	"put the money away in the safe on the — shelf"

C REAL ROBOT KEYFRAMES

We demonstrate the keyframes from two of our real robot tasks.

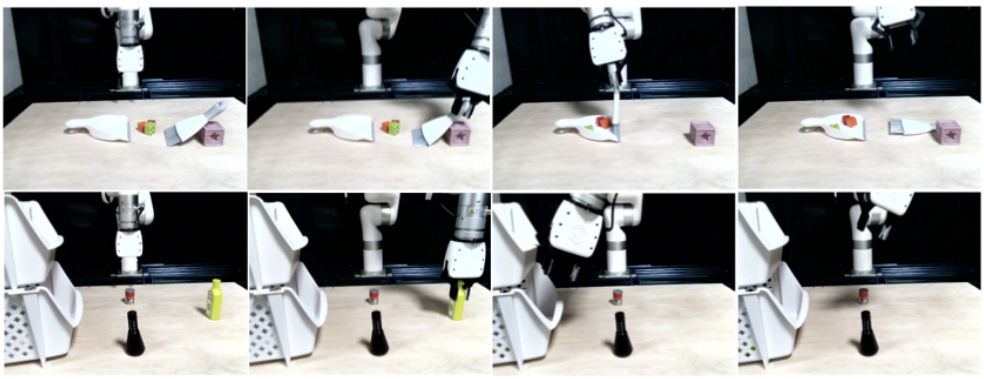

Figure 6: **Keyframes for Real-Robot tasks** We give two examples of keyframes used in our real robot tasks.

D MODEL ARCHITECTURES

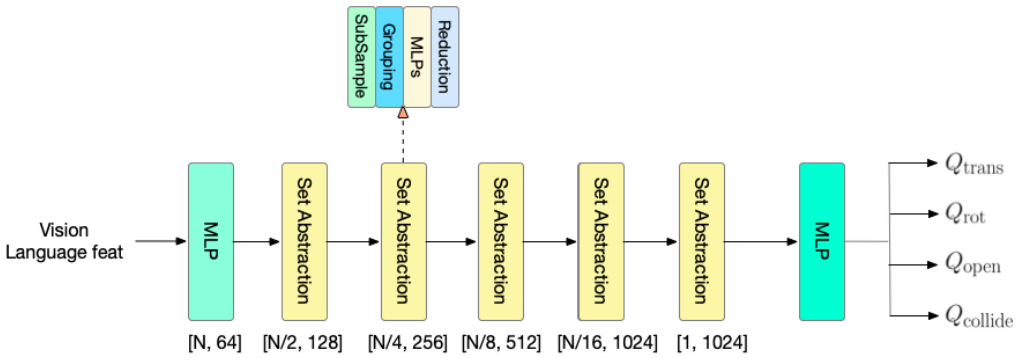

Figure 7: **Fusion Decoder Architecture.**

3D Encoder. We use a 3D UNet with 4.72M parameters to encode the input voxel $100^3 \times 10$ into a deep 3D volumetric representation of size $100^3 \times 64$.We provide the PyTorch-Style pseudo-code for the forward process as follows. Each conv layer comprises a single 3D Convolutional Layer, followed by Batch Normalization, and Leaky ReLU activation.

```
def forward(self, x):
   conv0 = self.conv0(x) # 100^3x8
   conv2 = self.conv2(self.conv1(conv0)) # 50^3x16
   conv4 = self.conv4(self.conv3(conv2)) # 25^3x32
   conv6 = self.conv6(self.conv5(conv4)) # 13^3x64
   conv8=self.conv8(self.conv7(conv6)) # 7^3x128
   x = self.conv10(self.conv9(conv8)) # 7^3x256
   x = conv8 + self.conv11(x) # 7^3x128
   x = conv6 + self.conv13(x) # 13^3x64
   x = conv4 + self.conv15(x) # 25^3x32
   x = conv2 + self.conv17(x) # 50^3x16
   x = self.conv_out(conv0 + self.conv19(x)) # 100^3x64
   return x
```

Fusion Decoder. We use several set abstraction blocks to fuse pre-trained 3D semantics feature, geometric point cloud feature, language feature from CLIP and robot proprioception embedding. This generates a vision-language feature of size 1024 as input of policy MLP.

Noise Predictor. The architecture of the noise predictor is a modified U-Net architecture designed to handle 1D inputs and incorporates conditional inputs. It's comprised of an Encoder, Decoder, and additional components to process conditional inputs.

- **Encoder:** The encoder is composed of a sequence of conditional residual block and each block has two "Conv1d $\rightarrow$ GroupNorm $\rightarrow$ Mish" components. A downsampling layer is applied after each block, performing downsampling with Conv1d.
- **Decoder:** Similar to the encoder while replacing the downsampling layers with upsampling layers performed by ConvTranspose1d.
- **Final Convolution Layer:** A sequence comprising a "Conv1d $\rightarrow$ GroupNorm $\rightarrow$ Mish" and Conv1d layer.

We apply the Feature-wise Linear Modulation (FiLM) (Perez et al., 2018) to enable the noise predictor to predict the noise with conditional input, the fused feature. This technique is particularly useful in conditional generation tasks. The FiLM module performs modulation by applying a simple affine transformation to each feature map. Given a feature map x, the FiLM transformation is defined as:

$$\text{FiLM}(x) = \gamma \cdot x + \beta, \tag{6}$$

where x is the input feature map, γ is the scale parameter, and β is the shift parameter. In our noise predictor architecture, the fused feature is used to predict the FiLM parameters γ and β. The predicted parameters are then used to modulate the feature maps within each block.

Policy MLP. The Policy MLP is composed of several MLPs. The translation output has one independent MLP and the remaining rotation, collision, and open action, share another set of MLP.