

## A Further Details on Model Architecture and Training

We use a standard U-Net (i.e. without and self-attentions or cross-attentions as recent works). We use 48 base channels with 4 downsample and 4 upsample blocks. In each block, there is 2 convolutional layers, each followed by a ReLU activation and a GroupNorm with a group size of 8. We use bilinear interpolation as our upsampling operations. For our proposed dual-head architecture, the two branches share all the submodules until the second last upsample block (i.e. the last two upsample blocks are separated). For its diffusion instantiation, we first use a Sinsoidal embedding followed by a 2-layer MLP to produce the time embedding, and this embedding is then broadcast and added to the UNet feature map for conditioning. For the concatenated intrinsics map, we apply normalization on the inverse focal length terms.

**Training Details (Common)** We use an AdamW optimizer to train our model with a learning rate of 0.0001 and a weight decay of 0.01. We use a batch size of 48 per GPU in both phase I and II training. For the phase I training, we use 8 A100 GPUs. For the phase II training, we use 1 A100 GPU and train the model for 1000 epochs.

**Training Details (Phase II)** For the phase II training, we also apply data augmentation to the observation. Specifically, the most important augmentation is the random brightness and hue applied to the RGB camera observation as we find the lighting can vary throughout the day. We do not apply random shift augmentation because this will alter the camera center and affect motion field prediction.

**Diffusion Model** For our diffusion model instantiation, we used a Denoising Diffusion Implicit Model (DDIM) as our noise scheduler. We use 100 training steps with `squared_cosine_v2` scheduling. During inference, we use 12 sampling steps without clipping denoised samples.

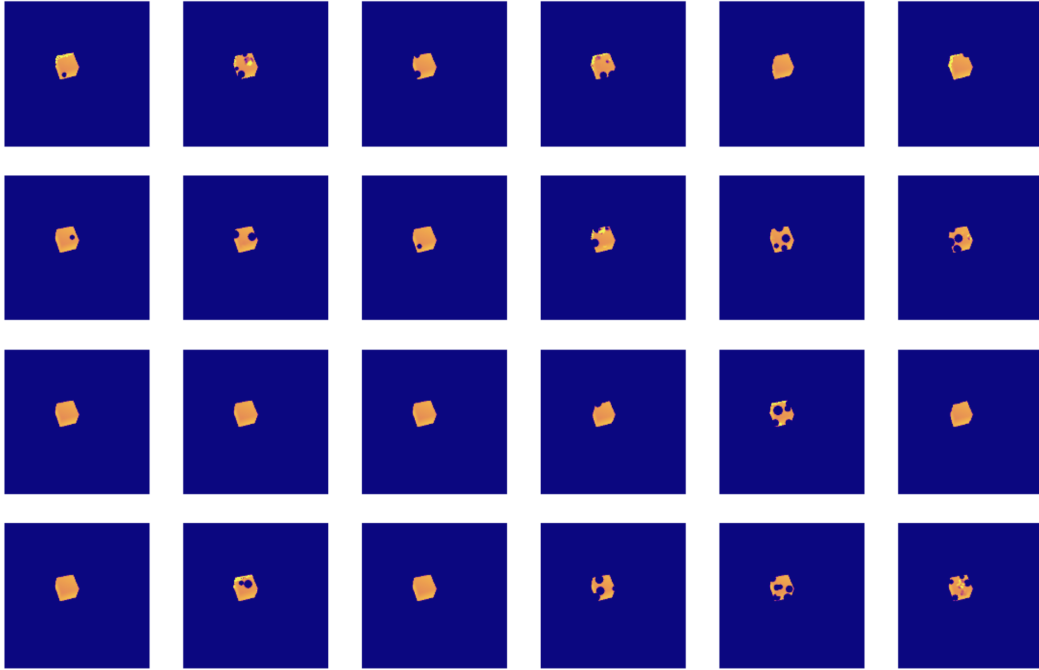


Figure 1: Example randomization over depth for a single sample. Each augmented sample is generated through a random sequence of atomic random augmentations.

## B Further Details on the Simulation Dataset

We visualize more randomization examples (Figure 1) in our training dataset. There are two types of noises: one is random masking (missing value), and the other one is random shift noise added to depth value. For the random noise, we introduce both spatially correlated and uncorrelated components. Notably, the uncorrelated noise is applied with much lower intensity, as our observations indicate that the majority of depth noise is spatially correlated. We also provide the visualization of each data

channel (Figure 2). We use a random subsampling for the 3D pixel flow. This can speed up the data labeling process (phase II-A) – we do not need a dense 3D pixel flow as we can produce motion field from a sparse 3D pixel flow.

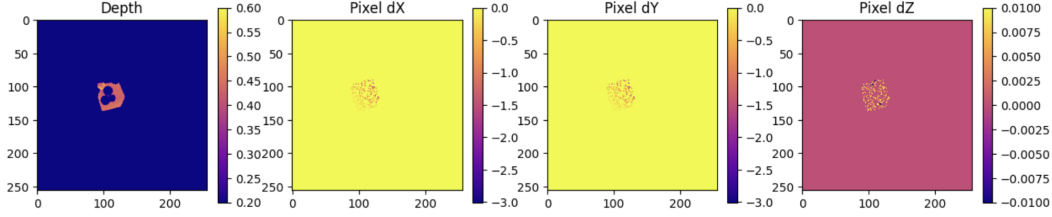


Figure 2: Phase I Motion Field Estimator Input. We use a sparse, randomly downsampled pixel flow.

## C Further Details on Real World Experiments

The number of real world demonstrations used by each experiment are shown in Table 1. As the demonstration is collected by a human hand, the average duration of each demonstration is short – only about 3-5s across each task. We visualize some of the human demonstration samples (i.e. camera observation) in Figure 4. During the robot deployment, the camera views the workspace at the same site.

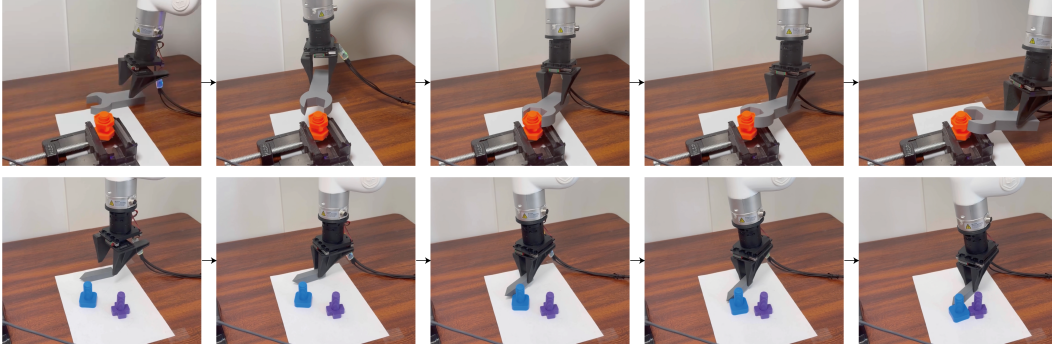


Figure 3: More Real World Rollout Results. See the video for the full process. Note that these images are not the camera observation for our policy.

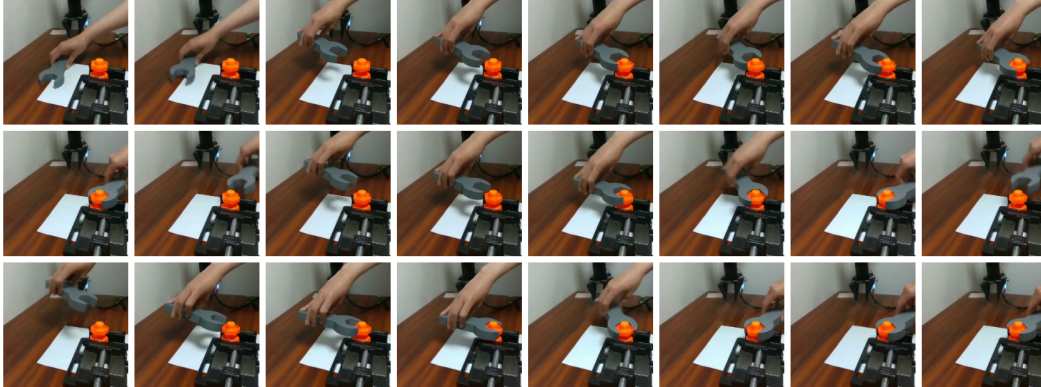


Figure 4: Example Realworld Video Demonstration by Intel D435 Camera.

We visualize more real world rollouts in Figure 3 and we also attach some video clips in the supplementary materials. For the precise manipulation task, the most common failure mode is the object misalignment due to small errors. However, we hypothesize that the error may be due to the tiny camera calibration error ( $1 \sim 2\text{mm}$ ) and it is hard to disentangle this effect. The second most

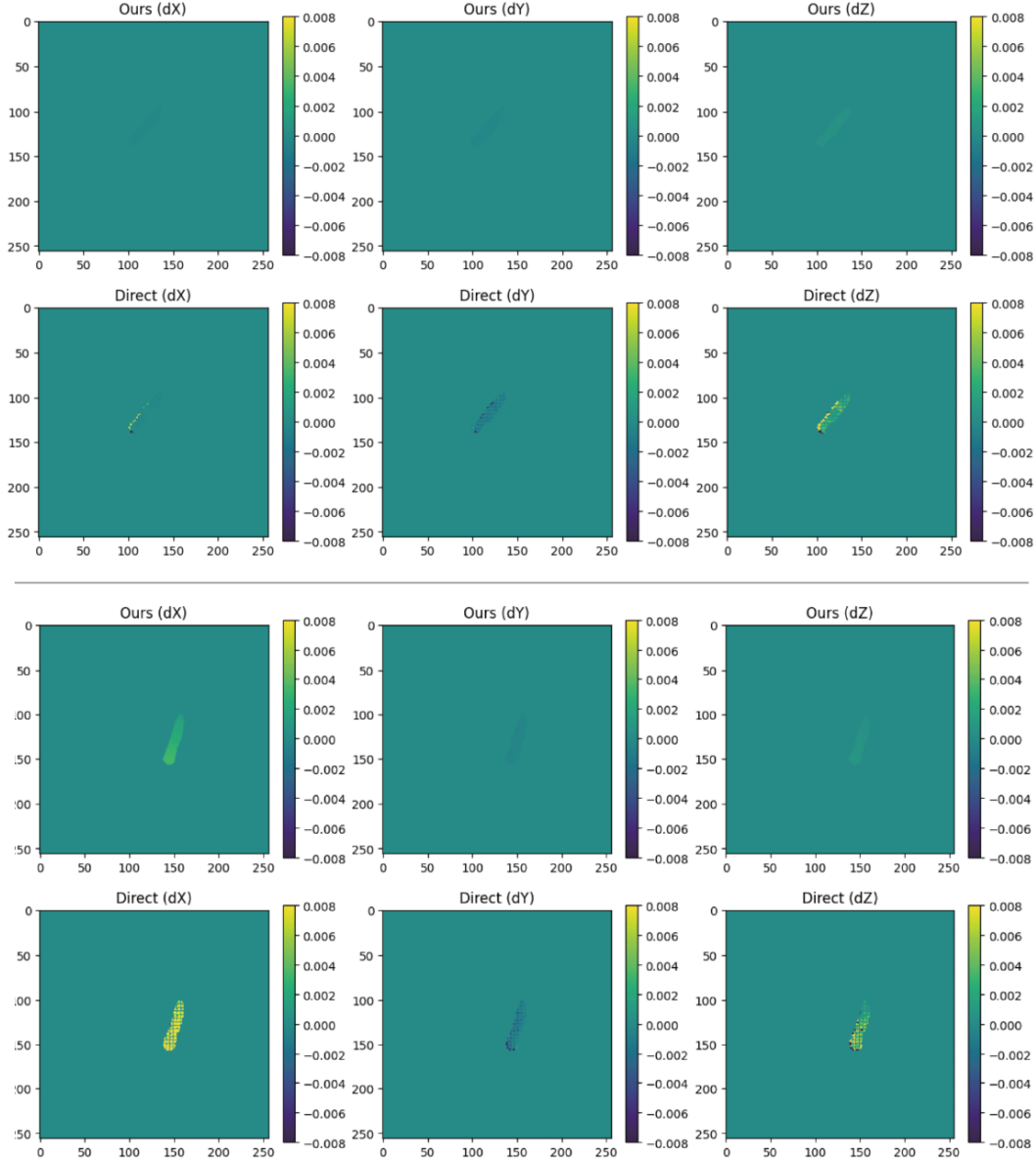


Figure 5: Motion Field Comparison (2cm-wide pen motion): Our method produces a smoother motion field than the direct method, which exhibits noticeable noise.

Task	T1 (Rotate)	T2 (Track)	T3 (Push)	T4 (Wrench)	T5 (Insert)
<b>Num Demo</b>	80	80	80	80	125

Table 1: Number of Human Video Demonstrations.

common failure pattern is the covariate shift problem caused by the rotation or translation error. In some extreme cases, the arm can push too hard against the base (the wrench task) and run into an out-of-distribution state.

Although the robot’s motion appears somewhat slow in the supplementary video, our analysis reveals that the primary bottleneck lies in the data transmission delay between the robot and the host. The inference system on the host itself operates efficiently at 8 to 12 Hz on a single A100 GPU (with a video-mode SAM2 for image preprocessing).