A. Appendix

The appendix is organized as follows:

- In Sec. A.1 we provide more details on the VLM pre-training including VQA tasks, encoder fusion strategies, 3D tokenization and data annotation pipeline.
- In Sec. A.2 we provide more details on the VLA training including data mixture, architecture, flow matching and design decision ablation results.
- In Sec. A.3 we provide the scoring rubrics for real-world evaluation tasks
- In Sec. A.4 we discuss the differences between Bridge V2 and DROID datasets for zero-shot control evaluations in unseen environments.

Dataset	Weight
austin_buds_dataset	0.5
austin_sailor_dataset	2.0
austin_sirius_dataset	0.5
berkeley_autolab_ur5	1.0
berkeley_cable_routing	0.1
berkeley_fanuc_manipulation	1.0
bridge	18.0
dlr_edan_shared_control	0.1
droid	35.0
fmb	1.5
fractal20220817_data	12.0
furniture_bench_dataset	1.5
iamlab_cmu_pickup_insert	0.3
kuka	4.0
language_table	1.5
nyu_franka_play_dataset	0.3
roboset (kinesthetic)	2.0
roboset (teleoperation)	5.0
roboturk	3.0
stanford_hydra_dataset	3.0
taco_play	2.0
toto	1.5
ucsd_kitchen_dataset	0.2
utaustin_mutex	3.0
viola	1.0

Table 4. Open X-Embodiment data mixture for SPEAR-1 pre-training

A.1. VLM training

A.1.1. VQA tasks for VLM pre-training

The Visual Question Answering (VQA) tasks used during VLM pre-training are inspired by VLA embodied tasks and aim to embed as much control-relevant 3D information into the VLM as possible. We use templated question-answer pairs grouped in the following categories:

- **3D keypoints prediction**: Output the 3D coordinates of the closest, furthest and center points of an object with respect to the camera frame.
- **3D bounding prediction**: Output the vertices of the 3D bounding box of an object.
- **Object-to-object distance prediction**: Output the direct distance between object X and object Y in 3D space as well as its xyz components.
- **Object-to-object bounding box prediction**: Output the distance between the bounding box vertices and the centers of object X and object Y.
- **Backprojection**: Locate the vertices of the 3D bounding box of an object on the 2D image.
- **Chain-of-thought comparisons**: What is the distance from the camera to object X? What is the distance from the camera to object Y? Which object is closer to the camera?

To further encourage the model to 'reason' over the information provided and attend to the right objects, in a single training example we use a random number (between 1 and 4) of question-answer pairs corresponding to different prompts and objects in the scene. To resolve ambiguities, if two instances of the same type of object appear in the image, we filter them out and never ask questions about them.

A.1.2. VLM encoder fusion strategies

We experimented with 2 different strategies to combine the outputs of the SigLIP and MoGe encoders:

1. Averaging the visual features predicted by both encoders after projecting each set of features through a separate linear layer to the LLM embedding space. In particular, for SigLIP we take only the tokens at the last layer of the vision encoder, while for MoGe we take the tokens at the last 4 layers of the encoder, following the approach used by MoGe architecture to decode the features to a 3D point cloud.
2. Using MoGe's predicted 3D point cloud $\mathbf{P}$ in the camera ego pose (in an affine-invariant space) and adding them to the SigLIP encoder features, similar to SpatialVLA [35]. In particular, MoGe's 3D point cloud output $\mathbf{P} \in \mathbb{R}^{H \times W \times 3}$ is embedded to $\mathbf{P}' \in \mathbb{R}^{h \times w \times d}$ through a projector $\psi(\cdot)$, composed of normalization, convolution, sinusoidal embedding $\gamma(x) = (x, \sin(2^0 \pi x), \cos(2^0 \pi x), \ldots, \sin(2^{L-1} \pi x), \cos(2^{L-1} \pi x))$ [30] and an MLP. Finally, the features $\mathbf{F}' = \mathbf{F} + \mathbf{P}'$ are fed to PaliGemma's SigLIP linear projector, where $\mathbf{F} \in \mathbb{R}^{h \times w \times d}$ denotes the features at the SigLIP encoder output.

During our preliminary VLM evaluations we found the first strategy to demonstrate qualitatively better performance on bounding box prediction tasks. In particular, models trained with the second approach struggled to consistently output "grammatically" correct bounding boxes, *e.g.* they would output 22 or 23 3D tokens instead of the required 24. We therefore used the first approach for all

Dataset	Domain / Subset	# Annotated Images	Segmentation Masks
EgoExo4D [13]	Cooking & Bike Repair	∼200k	GT
Bridge [45]	Robot Demonstrations	∼30k	SAM2 Generated
Total		∼230k	

Table 5. Annotated image counts for training dataset construction, with segmentation mask availability.

Task	0.25	0.50	0.75	1.00
Carrot on Plate (w/ distractors & elevations)	Reach carrot	Pick up carrot	Drop on/near plate	Correctly place on plate
Marker in Cup (w/ distractors)	Reach marker	Pick up marker	Drop on/near cup	Place inside cup
Cover the Pot	–	Pick up lid	Drop lid on pot	Correctly cover pot
Apple in drawer and close	Pick up the apple	Put the apple in the drawer	Half-close the drawer	Fully close the drawer

Table 6. Scoring rubric for Franka evaluation tasks.

Task	0.25	0.50	0.75	1.00
Eggplant in pot	Reach the eggplant	Pick up the eggplant	Drop the eggplant near the pot	Drop the eggplant on the pot
Pink cup on blue plate	Reach the pink cup	Pick up the pink cup	Drop the pink cup near the plate	Place the pink cup correctly
Chess piece on board	-	Go to the brown chess piece	Pick up the brown chess piece	Drop it on the board
Lobster in the pan	-	Pick up the lobster	Drop the lobster near the pan	Place the lobster inside the pan
Corn between cups	-	Pick up the corn	-	Place the corn between the cups

Table 7. Scoring rubric for WidowX evaluation tasks.

VLM pre-training experiments in the main paper.

A.1.3. 3D tokenization

To encode 3D information into text we extend the PaliGemma tokenizer with $N = 1024$ 3D tokens, as 3D coordinates are conceptually different from the existing visual and language tokens. This is in line with PaliGemma's approach of extending Gemma's tokenizer to pixel locations. Each 3D token corresponds to a quantized *distance value* in the range $[z_{\min}, z_{\max}]$, where $z_{\min}$ and $z_{\max}$ are computed as the 1st and 99th quantiles of the 3D point cloud distribution along any of the xyz coordinates.

We found the *distance values* in the data to approximately follow a Normal distribution. Therefore, to allow for more accurate tokenization, we compute non-uniform bins with fine-grained discretization around the mean and spread out widths near the tails such that the distribution of 3D tokens is approximately uniform.

We initialize the new token embedding weights from a multivariate normal distribution that has the mean and covariance of the pretrained embeddings [15, 16].

A.1.4. VQA data annotation pipeline

We follow the method described in Section 3.1 in order to enrich 2D images with semantics, segmentation masks and 3D point clouds. We also experimented with GroundingDINO [29] instead of Gemini, but we found the semantic labels produced by GroudingDINO to be a lot less accurate and consistent. We found that prompting SAM2 [36] with 2D bounding boxes near the target objects, leads to segmentation masks of high quality.

We also found that MoGe [47] outputs depths at different scales depending on the input image size. Therefore, we resize all our images to 840x630 for MoGe point cloud annotations.

For 3D bounding box estimation, after filtering the 3D point cloud with a segmentation mask, we run statistical outlier removal and esitmate an oriented 3D bounding box around the remaining points using Open3D [52]. To facilitate learning, we order all 8 bounding box vertices in a consistent way, starting based on their spatial coordinates with respect to the camera frame.

A.2. VLA training

A.2.1. Data mixture

We report the VLA training data mixture in Tab. 4. The sampling weights are chosen manually based on dataset size, visual and task diversity, and quality of language annotations.

(a) Training environment: Toykitchen 2 (b) SIMPLER: Put Carrot on Plate (c) SIMPLER: Put Eggplant in Yellow Basket (d) SIMPLER: Put Spoon on Towel (e) SIMPLER: Stack Green Block on Yellow Block

Figure 6. **3D ablation environments on WidowX.** (a) Training data subset from Bridge V2 [45]. (b) - (e) SIMPLER evaluation environments.

A.2.2. VLA training details

Reference Frames. In this work we focus on learning position control of fixed-base single-arm manipulators. Each control in the sequence $\mathbf{A_t} = [\mathbf{a}_t, \mathbf{a}_{t+1}, \dots \mathbf{a}_{t+H-1}]$ is defined as a delta with respect to the current end-effector cartesian pose $\Delta_{EE} = [\Delta_T, \Delta_R]$. The translation component, Δ_T is in robot base frame and the rotation component, Δ_R, is in end-effector frame. The gripper action is binary.

Action Chunking. During VLA training we use an action chunk of size $H = 5$ and frequency of 5Hz. As not all datasets in Open X-Embodiment provide action labels at 5Hz, we downsample or upsample the actions accordingly via linear interpolation. This is done with the goal to encourage the model to share knowledge across datasets with different control frequencies and embodiments instead of 'memorizing' each dataset separately.

Architecture. Similar to π_0 [5], SPEAR-1 combines a VLM, which processes the image-language inputs, with an *action expert* module, which processes robot proprioception observations and predicts the robot action sequence conditioned on the VLM transformer's intermediate key-value pairs. The action expert has the same architecture and number of layers as the Gemma [42] transformer and configuration downsized to $token\_size = 4096, hidden\_size = 4096$, for a total of ∼300M parameters, which is exactly the same as π_0 [5]. Corresponding layers in the VLM and the action expert have a shared attention operation with block-wise causal attention over the blocks $[\mathbf{I}_t, \mathbf{l}_t], [\mathbf{p}_t], [\mathbf{a}_{t+1}, \dots, \mathbf{a}_{t+H-1}]$. Within each block, there is full bidirectional attention and the tokens in each block can attend to tokens in previous blocks, but cannot attend to the tokens in future blocks. During training, only the action sequence prediction is supervised and gradient updates are propagated back to the VLM parameters through the shared attention layers.

A.2.3. Flow matching details

To address the inherent double coverage of 3D rotations by the unit quaternion group $\mathbb{S}^3$, we ensure that all quaternions used during training and inference lie in the same half-space defined by $\Re(\mathbf{q}) = \mathbf{q}_w > 0$.

Quaternion integration. Given a unit quaternion $\mathbf{q}_t \in \mathbb{S}^3$ and its time derivative $\dot{\mathbf{q}}_t \in \mathbb{R}^4$, we can compute the angular velocity of rotation via $\boldsymbol{\omega}_t = 2.0 \cdot \Im\mathfrak{m}(\mathbf{q}_t^* \otimes \dot{\mathbf{q}}_t) \in \mathbb{R}^3$. For a small time step Δt, the corresponsing delta rotation is given by a rotation vector around the unit axis $\boldsymbol{\omega} = \boldsymbol{\omega}/||\boldsymbol{\omega}||$ over an angle $\Delta\phi = ||\boldsymbol{\omega}||\Delta t$. The corresponding delta quaternion is given by

$$\Delta\mathbf{q} = \left[\cos\left(\frac{\Delta\phi}{2}\right), \boldsymbol{\omega}\sin\left(\frac{\Delta\phi}{2}\right)\right]. \tag{6}$$

The integrated unit quaternion is then given by $\mathbf{q}_{t+\Delta t} = \mathbf{q}_t \otimes \Delta\mathbf{q} \in \mathbb{S}^3$,

Rotation losses. The denoising vector field for quaternions $\mathbf{u}_t(\mathbf{q}_t^\tau|\mathbf{q}_t) \in \mathbb{R}^4$ is computed as:

$$\begin{aligned}\mathbf{u}_t(\mathbf{q}_t^\tau|\mathbf{q}_t) &= \frac{d\mathbf{q}_t^\tau}{d\tau} = \\ &= \frac{\theta}{\sin\theta}\left[-\cos\big((1-\tau)\theta\big)\,\mathbf{q}_\epsilon + \cos\big(\tau\theta\big)\,\mathbf{q}_t\right].\end{aligned} \tag{7}$$

The cosine loss is applied directly on the velocity predictions and has the form:

$$\mathcal{L}_t^{\cos}(\theta) = 1 - \mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t)[\mathbf{q}] \cdot \mathbf{u}(\mathbf{A}_t^\tau|\mathbf{A}_t)[\mathbf{q}]. \tag{8}$$

The geodesic loss is applied on an integrated rotation prediction $\mathbf{q}_{\theta,t}^{\tau+\delta} \in \mathbb{S}^3$, derived by integrating the noised input quaternion $\mathbf{q}_t^\tau$ with the predicted rotation velocity $\mathbf{v}_\theta(\mathbf{A}_t^\tau, \mathbf{o}_t)[\mathbf{q}]$ over a small time step δ. We follow the integration method described above. The target is given by the ground truth interpolated quaternion at time $t + \delta$, denoted as $\mathbf{q}_t^{\tau+\delta} \in \mathbb{S}^3$. The closed form geodesic loss is given by:

$$\mathcal{L}_t^{\text{geo}}(\theta) = \min|\mathbf{q}_t^{\tau+\delta} \pm \mathbf{q}_{\theta,t}^{\tau+\delta}|. \tag{9}$$

The complete rotation loss is given by:

$$\mathcal{L}_{\mathbb{S}^3}(\theta) = \sum_{k=t}^{t+H}\left[\mathcal{L}_k^{\cos}(\theta) + \mathcal{L}_k^{\text{geo}}(\theta)\right]. \tag{10}$$

A.2.4. VLA design decisions details

We present more details and results on the design choices we explored for VLA training.

Experiment	Put Carrot on Plate	Put Eggplant in Yellow Basket	Put Spoon on Towel	Stack Green Block on Yellow Block	Avg. Success Rate
224×224	**70.8%**	70.8%	79.1%	8.3%	57.25%
280×210	62.5%	**75.0%**	**83.3%**	**12.5%**	**58.3%**

Table 8. **Image resultion ablations.** Different resolutions lead to comparable results on SIMPLER WidowX tasks.

Experiment	Put Carrot on Plate	Put Eggplant in Yellow Basket	Put Spoon on Towel	Stack Green Block on Yellow Block	Avg. Success Rate
trainable SigLIP	**75.0%**	**100.0%**	79.1%	**37.5%**	**72.9%**
frozen SigLIP	62.5%	54.1%	83.3%	25%	56.3%
frozen-trainable SigLIP	66.6%	83.3%	**100.0%**	33.3%	70.8%
lower lr SigLIP	58.3%	58.3%	79.1%	29.1%	56.3%

Table 9. **Vision encoder training.** Trainable SigLIP outperforms other strategies on SIMPLER WidowX tasks. Frozen SigLIP followed by switching on gradients is comparable.

Experiment	Put Carrot on Plate	Put Eggplant in Yellow Basket	Put Spoon on Towel	Stack Green Block on Yellow Block	Avg. Success Rate
99-th quantile	54.1%	79.1%	79.1%	**33.3%**	61.5%
min-max const	**66.6%**	**87.5%**	**87.5%**	20.8%	**65.6%**
mean-std	45.8%	79.1%	45.8%	25.0%	49.0%

Table 10. **Translation controls normalization.** Normalizing translation controls with min-max constants outperforms other strategies on SIMPLER WidowX tasks.

Flow matching	Velocity Loss	Rotation loss	Put Carrot on Plate	Put Eggplant in Yellow Basket	Put Spoon on Towel	Stack Green Block on Yellow Block	Avg. Success Rate
linear	MSE	geodesic	41.6%	**100.0%**	41.6%	16.6%	50.0%
linear	cos	geodesic	41.6%	87.5%	50.0%	29.1%	52.1%
$\mathbb{S}^3$	MSE	geodesic	**45.8%**	62.5%	**75.0%**	**45.8%**	57.3%
$\mathbb{S}^3$	cos	geodesic	**45.8%**	79.1%	66.6%	**45.8%**	**59.4%**

Table 11. **Linear vs $\mathbb{S}^3$ Flow Matching for rotations.** $\mathbb{S}^3$ flow matching consistently outperforms linear flow matching on SIMPLER WidowX tasks.

Image Resolution. Image resolution ablations are presented in Tab. 8. We observe that square vs 4:3 aspect ratio does not significantly affect performance.

Fine-tuning vision encoders. Ablations on fine-tuning vision encoders are presented in Tab. 9. Trainable SigLIP strongly outperforms a frozen SigLIP as well as SigLIP with a lower learning rate compared to the rest of the weights. Freezing SigLIP and fine-tuning for additional 2k steps (frozen-trainable SigLIP) leads to comparable performance to trainable SigLIP, but requires an additional hyperparameter tuning.

Controls normalization. Ablations on translation controls normalization are presented in Tab. 10. Mean-std normalization is significantly worse than other forms of normalization. Min-max normalization with const values is slightly better than per-dataset min-max normalization with 1st and 99th quantiles.

Rotations. Partial ablations on rotation representations are presented in Tab. 11. $\mathbb{S}^3$ flow matching consistently outperforms linear flow matching. Cosine distance leads to slightly better performance than MSE for rotation velocity prediction.

A.2.5. 3D ablation details

SIMPLER WidowX experiments. For SIMPLER [22] 3D ablations on WidowX, we train on a subset of the Bridge V2 [45] dataset, containing demonstrations only from a single kitchen sink environment. The resulting subset comprises $\sim 41\%$ of the original Bridge V2 dataset. We train each VLA for 30k steps with batch size 512. Example images from the training and evaluation environments are shown in Fig. 6.

Franka experiments. For the 3D ablations on a real-world Franka robot, we train on the entire DROID [18] dataset for 100k steps with batch size 2048. Both models take as input both side and wrist cameras.

A.3. Real-world robot task description and scoring

We provide the detailed task progression scoring for all real-world evaluations on the WidowX and Franka in Tab. 6 and Tab. 7 respectively.

A.4. Zero-shot control: Bridge V2 vs. DROID

Model	Zero-shot control embodiments in real-world **unseen** environment
RT-1-X [32]	WidowX
RT-2-X [32]	WidowX, Google Robot
Octo [31]	WidowX, Google Robot**?**
OpenVLA [19]	WidowX
SpatialVLA [35]	WidowX
CogACT [21]	WidowX
FLOWER [38]	WidowX
MotoVLA [40]	WidowX
CoT-VLA [51]	WidowX
π_0 [5]	**Franka**, Others**?**
π_0**-FAST** [33]	**Franka**, Others**?**
$\pi_{0.5}$ [6]	**Franka**, Mobile Fibocom, Mobile Galaxea, Others**?**
Gemini Robotics 1.5	Bimanual Franka, Aloha, Apollo humanoid
RDT1	Bimanual UR5, Aloha
RDT2	Bimanual UR5, Bimanual Franka
SmolVLA	S0101
GROOT-N1.5	None
SPEAR-1 (ours)	WidowX, **Franka**

Table 12. Most works on generalist models for robot manipulation evaluate zero-shot control on Bridge V2 + WidowX using in-distribution environments. Only few do so on DROID + Franka in **unseen** environments.

Most works on generalist models for robot manipulation [19, 21, 32, 49, 51] evaluate the zero-shot control capabilities of their policies by pretraining on the Bridge V2 dataset [45] and deploying on the WidowX robot in environments close to the training distribution. Bridge V2, however, is not very diverse in the number of environments, objects,

and camera viewpoints. As a result, we observe that models pre-trained on Bridge V2 only perform well on WidowX environments when the deployment scenario is similar to what is seen in the dataset (e.g. in the blue toy sink environment), but are usually very sensitive to variations in camera position and OOD backgrounds and objects. In addition, the WidowX arm has a very low payload and short reach, which makes it unable to manipulate objects beyond the items in a toy kitchen set. The DROID dataset [18], on the other hand, is significantly more diverse, and the Franka arm used for data collection is more capable. Furthermore, DROID demonstrations are collected primarily in real-world environments instead of toy environments, the number of unique scenes is $20\times$ higher, and the camera viewpoints vary significantly. Therefore, we posit that pretraining on DROID and deploying on Franka is a superior experimental setup to showcase generalization to more realistic real-world scenarios, as shown by [2]. The diversity and richness of DROID, however, is at the same time a challenge. Training generalist control policies on DROID that perform well zero-shot on a Franka robot in unseen environment, is a task that, to the best of our knowledge, has been tackled successfully only by a handful of works so far [6, 33]. In contrast, multiple other works that pre-train on DROID, resort to mixing or fine-tuning on demonstrations collected from the specific target environment in order to achieve good performance [19, 21, 35, 38, 51]. Therefore, as suggested also by [33], we argue that achieving state-of-the-art performance on zero-shot control on the DROID setup by pre-training on DROID is a significantly stronger result than pre-training on Bridge V2 and deploying on WidowX.