

1	Appendix	
2	Contents	
3	A Algorithm Pseudocode	2
4	B Network Architecture	2
5	B.1 Real World Experiments	2
6	B.1.1 Observation Encoder: f_ϕ	2
7	B.1.2 Policy: π_θ	3
8	B.2 Sim PushT	4
9	C Baselines	5
10	C.1 Real World Experiments	5
11	C.2 Sim PushT	6
12	D Robot Data	7
13	E Embodied human data	7
14	F Data processing and alignment.	8
15	G Additional Task Details	8
16	G.1 Real World Experiments	8
17	G.2 Sim PushT	9
18	H Supplementary Simulation Results	10
19	I Pseudo-pair Visualisation	10
20	J Latent Space Visualisation	11

21 A Algorithm Pseudocode

22 **Overview.** Algorithm 1 describes the joint policy co-training of the human and robot data with the
23 joint OT loss.

24 **Mini-batch sampling.** At each iteration we draw two size- B mini-batches, one from the human
25 dataset \mathcal{D}_H and one from the robot dataset \mathcal{D}_R (Lines 3–4).

26 **Shared encoder.** Both batches are fed through a *shared* encoder f_ϕ that maps raw observations o to a
27 latent embedding $z \in \mathbb{R}^d$ (Line 6).

28 **Behavioural pairing via DTW.** The Dynamic Time Warping (DTW) cost is applied on the *action*
29 sequences $a_H^{(i)}$ and $a_R^{(j)}$ from both domains. The assignment $i^*(j)$ (Line 11) identifies a single
30 pseudo-pair in \mathcal{D}_H for each robot trajectory j which minimizes the DTW cost.

31 **Sinkhorn.** The Sinkhorn-Knopp algorithm is used to compute an entropy-regularized Optimal
32 Transport plan from the uniform probability mass on the minibatch supports.

33 **Joint-OT.** Latent distances $D_{ij} = \|z_H^{(i)} - z_R^{(j)}\|_2^2$ are discounted by $\lambda < 1$ when (i, j) is behaviourally
34 matched (Lines 12–17), producing a shaped cost \tilde{C} that biases the OT loss computed from the
35 Sinkhorn OT transport plan.

36 **Behaviour cloning and objective.** The decoder π_θ produces domain-specific actions and is trained
37 with a supervised loss $\mathcal{L}_{\text{BC-cotrain}}$ (Lines 22–24). The overall objective (Line 26)

$$\mathcal{L} = \mathcal{L}_{\text{BC-cotrain}} + \alpha \mathcal{L}_{\text{OT-joint}}$$

38 balances imitation fidelity with latent-space alignment, with an α balancing factor between the two
39 losses.

40 B Network Architecture

41 The following section describes how f_ϕ and π_θ are parameterized in the simulation and real-world
42 experiments. All hyperparameters are summarized in Table 1, Table 2 for real-world and in Table 3
43 for simulation.

44 B.1 Real World Experiments

45 B.1.1 Observation Encoder: f_ϕ

46 f_ϕ consists of two components, "stems" for each observation modality and a "trunk". The stems are
47 shallow networks that encode heterogenous input spaces into a fixed representation to allow joint
48 learning between human and robot observations.

49 **Vision Stem.** Given an RGB frame $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, we normalize using ImageNet normalization and
50 then pass it through a ResNet-18 encoder truncated *before* the global-pool layer to obtain a $7 \times 7 \times 512$
51 feature map. The 49 spatial patches are flattened and projected with a single-layer MLP of hidden
52 dimension d_{proj} , producing 49 tokens of size d_{proj} per image. A single multi-head cross-attention
53 block (D_{stem} heads, per-head width d_{attn}) employs L learnable query tokens of dimension d that
54 attend to these 49 patch tokens; the resulting L query outputs constitute the vision tokens passed to
55 the shared trunk.

56 **Proprioceptive Stem.** The proprioceptive observation vector $\mathbf{q} \in \mathbb{R}^{d_q}$ (joint angles, end effector
57 pose etc.) is z-score normalized and passed through a single-layer MLP of hidden dimension d_{proj} ,
58 producing one token $k_{\text{proprio}} \in \mathbb{R}^{d_{\text{proj}}}$. A single multi-head cross-attention block (D_{stem} heads, per-
59 head width d_{attn}) uses the L learnable query tokens as in the vision branch to attend to this key/value
60 token; the resulting L query outputs constitute the proprioceptive tokens that are forwarded to the
61 shared trunk.

62 **Trunk.** The *shared* "trunk" is a standard multi-block transformer encoder, which consists of D_{trunk}
63 heads, N_{trunk} blocks, and an embedding dimension of d . Each head has an attention dimension of
64 d/D_{trunk} . A token sequence of length $L \cdot m$ by concatenating the token outputs of m observation
65 encoding stems. A set of M learnable context tokens are prepended to the token sequence. The
66 new sequence of $M + m \cdot L$ are input into the trunk. The first M are extracted from the output
67 sequence and represent the feature output z of f_ϕ , and consequently where $\mathcal{L}_{\text{OT-joint}}$ is applied.

Algorithm 1 EgoBridge Co-Training for Human \leftrightarrow Robot Imitation

Require: Human demos \mathcal{D}_H , robot demos \mathcal{D}_R , OT loss weight α , DTW discount λ , Sinkhorn regularization ε , LR schedule η

Ensure: Encoder f_ϕ and policy decoder π_θ

- 1: Initialize network parameters ϕ, θ
- 2: **while** training not converged **do**
- 3: $\mathcal{B}_H = \{o_H^{(i)}, a_H^{(i)}\}_{i=0}^B \leftarrow \text{SAMPLEBATCH}(\mathcal{D}_H, B)$ \triangleright sample human data
- 4: $\mathcal{B}_R = \{o_R^{(i)}, a_R^{(i)}\}_{i=0}^B \leftarrow \text{SAMPLEBATCH}(\mathcal{D}_R, B)$ \triangleright sample robot data
- 5: $z_H \leftarrow f_\phi(o_H), z_R \leftarrow f_\phi(o_R)$ \triangleright encode raw observation batch into shared latent space
- 6: **for all** $i \in \{0, \dots, |B_H|-1\}, j \in \{0, \dots, |B_R|-1\}$ **do**
- 7: $A_{ij} \leftarrow \text{DTW}(a_H^{(i)}, a_R^{(j)})$ \triangleright DTW distance between actions
- 8: **end for**
- 9: $i^*(j) \leftarrow \arg \min_i A_{ij} \forall j \in \{0, \dots, |B_R|-1\}$ \triangleright pick the most similar human traj. for each robot traj.
- 10: **for all** $i \in \{0, \dots, |B_H|-1\}, j \in \{0, \dots, |B_R|-1\}$ **do**
- 11: $D_{ij} \leftarrow \|z_H^{(i)} - z_R^{(j)}\|_2^2$ \triangleright observation feature cost
- 12: **if** $i = i^*(j)$ **then**
- 13: $\tilde{C}_{ij} \leftarrow \lambda D_{ij}$ \triangleright discount cost if DTW says the pair matches
- 14: **else**
- 15: $\tilde{C}_{ij} \leftarrow D_{ij}$
- 16: **end if**
- 17: **end for**
- 18: $\mu_H \leftarrow \frac{1}{|B_H|} \mathbf{1}, \mu_R \leftarrow \frac{1}{|B_R|} \mathbf{1}$ \triangleright uniform probability mass on the minibatch supports
- 19: $T^* \leftarrow \text{SINKHORN}(\mu_H, \mu_R, \tilde{C}, \varepsilon)$ \triangleright compute differentiable OT loss with entropic reg.
- 20: $\mathcal{L}_{\text{OT-joint}} \leftarrow \sum_{i,j} T_{ij}^* \tilde{C}_{ij}$
- 21: $\hat{a}_H \leftarrow \pi_\theta(z_H), \hat{a}_R \leftarrow \pi_\theta(z_R)$ \triangleright predict actions for both domains
- 22: $\mathcal{L}_{\text{BC-cotrain}} \leftarrow \mathcal{L}_{\text{BC}}(\hat{a}_H, a_H) + \mathcal{L}_{\text{BC}}(\hat{a}_R, a_R)$ \triangleright behaviour-cloning loss across domains
- 23: $\mathcal{L} \leftarrow \mathcal{L}_{\text{BC-cotrain}} + \alpha \mathcal{L}_{\text{OT-joint}}$
- 24: $(\phi, \theta) \leftarrow \text{OPTIMIZERSTEP}(\phi, \theta, \nabla_{\phi, \theta} \mathcal{L}, \eta)$
- 25: **end while**
- 26: **return** f_ϕ, π_θ

68 B.1.2 Policy: π_θ

69 π_θ is conditioned on the output z of f_ϕ and is parameterized by a DETR-style multi-block transformer
70 decoder "head" [1].

71 **Head.** The head consists of N_{head} blocks and D_{head} heads, a cross-attention dimension of
72 $d_{\text{head}} \cdot D_{\text{head}}$ and a self-attention dimension of $d_{\text{head}}/D_{\text{head}}$. This hybrid attention is designed to
73 promote increased conditioning from the latent z . k learnable tokens of d_{head} are input into the
74 model, where K corresponds to the action chunk length k . Each block in the decoder consists of
75 alternating self-attention on the k input tokens, and cross-attention with the context M tokens (z)
76 from the trunk. After the final block, a linear layer projects the output to d_a corresponding to the
77 cartesian action dimension, which is described precisely in Section D. Since the human data only
78 comprises of 3D position, the loss is a masked loss which is only calculated on the values in the
79 prediction corresponding to the 3D position for human data. The loss \mathcal{L}_{BC} on human data is Smooth
80 L1 loss on the 3D position, while the loss \mathcal{L}_{BC} on robot data is Smooth L1 loss on the 3D position,
81 gripper value and MSE loss on the orientation.

82 **Complete Architecture.** The network consists of a *shared* vision stem processes main egocentric
83 RGB images ($I_{\text{ego}} \in \mathbb{R}^{H \times W \times 3}$) from both human and robot and produces L tokens. Followed by
84 this are individual *embodiment-specific* proprioceptive stems that map q^H and q^R proprioceptive
85 features to an additional L tokens. To the vision and proprioceptive tokens, left and right wrist
86 RGB images ($I_{\text{wrist}} \in \mathbb{R}^{H \times W \times 3}$) are mapped to additional tokens by one additional wrist vision
87 stem for single-arm robot data and two additional wrist vision stems for bimanual robot data. M
88 learnable tokens are prepended to the token sequence and passed to the trunk. The OT-joint loss

Table 1: Hyperparameters for Real-World Experiments

Symbol	Value
$H \times W \times C$	$480 \times 640 \times 3$
d_{proj}	256
D_{stem}	8
d_{attn}	64
L	16
$d_q(\text{Robot})$	7 or 14 (joint positions)
$d_q(\text{Human})$	6 or 12 (end effector pose as xyz + euler)
D_{trunk}	8
N_{trunk}	16
d	256
M	8
D_{head}	8
N_{head}	8
d_{head}	64
k	100
d_a	7 or 14 (xyz + euler angles + gripper position)
$L_{BC}(\text{Robot})$	$\text{SmoothL1}(\text{xyz}) + \text{SmoothL1}(\text{gripper}) + 0.5 \cdot \text{MSE}(\text{euler})$
$\mathcal{L}_{BC}(\text{Human})$	$\text{SmoothL1}(\text{xyz})$
$\mathcal{L} = \mathcal{L}_{BC-\text{cotrain}} + \alpha \mathcal{L}_{\text{OT-joint}}$	$\alpha = 0.7$

is applied on the first M tokens of the output. The same tokens are decoded by the head into $a_{t:t+k} \in SE(3) + \text{grripper}$, where an action for a single arm $\in R^{k \times 7}$ with the orientation expressed as euler angles in the yaw-pitch-roll convention. For bimanual tasks, the action comprises of two such action trajectories concatenated resulting in an output $\in R^{k \times 14}$.

Training Details. We train the real world EgoBridge model on a single L40s gpu for 100000 iterations on the Drawer task, 110000 iterations on the Laundry task, and 120000 iterations on the Scoop Coffee task, which takes about 24 hours. More details are in Table 2.

Table 2: Training Details for Real World Experiments

Parameter	Value
Optimizer	AdamW
Learning Rate	5×10^{-5}
Weight Decay	0.0001
Scheduler	Linear
Batch Size	32
Data Augmentations	ColorJitter + ImageNet Normalization (ResNet)
OT Loss	GeomLoss [2]
Blur	0.05
Distance	Sinkhorn

B.2 Sim PushT

Data Processing. We follow the PushT environment setup introduced in the Diffusion Policy benchmark suite [3]. Each demonstration comprises a sequence of RGB observations and associated low-dimensional proprioception, along with corresponding action labels. The RGB observations $I_t \in \mathbb{R}^{96 \times 96 \times 3}$ are normalized using ImageNet statistics. The proprioception consists of 2D end-effector positions (x, y) , which are min-max normalized per-dimension using statistics computed over the entire dataset. The action at each timestep is defined as the target 2D position of the end-effector, also normalized with per-dimension min-max normalization. Action chunks of length $k = 16$ are extracted from each demonstration trajectory. No observation history is used in the input, i.e., the observation context window is 1.

Encoder (f_ϕ). We employ the same vision encoder architecture as in the original Diffusion Policy paper [3], which uses a truncated ResNet-18 to extract spatial feature maps from $96 \times 96 \times 3$ RGB images. The proprioceptive input (normalized x, y position) is concatenated channel-wise to the image feature maps before being passed as the global conditioning to the UNet-based diffusion decoder. This architecture has been shown to effectively fuse visual and proprioceptive features for policy learning in low-dimensional manipulation tasks like PushT. The Joint-OT loss is applied on the image-proprio concatenated feature output.

Policy (π_θ). The policy architecture mirrors the conditional diffusion policy design [3]. The output is a denoising network trained to generate a trajectory segment of length $k = 16$ for 2D end-effector coordinates in normalized space. During training, the policy receives a noisy version of the ground-truth action chunk, and learns to iteratively denoise the sample using the concatenated latent observations. The diffusion loss is the weighted MSE over all denoising steps in the sampling process. During inference, the policy performs iterative denoising starting from Gaussian noise to generate 2D trajectory samples conditioned on the current observation.

Training Details. We train the simulation EgoBridge model on a single A40 GPU for 130000 iterations, which corresponds to around 2 hours of training time. The hyperparameters and training details are summarized in Table 3.

Table 3: Training Details for Simulation Experiments (PushT)

Parameter	Value
Optimizer	AdamW
Learning Rate	1×10^{-4}
Weight Decay	1×10^{-6}
Scheduler	Cosine
Warmup Steps	500
Iterations	130,000
Batch Size	32
Exponential Moving Average (EMA)	Power = 0.75
Data Augmentations	ImageNet Normalization (ResNet)
\mathcal{L}_{BC} (Triangle & Circle)	$MSE(xy)$
$\mathcal{L} = \mathcal{L}_{BC-cotrain} + \alpha \mathcal{L}_{OT-joint}$	$\alpha = 0.2$
OT Loss	GeomLoss [2]
Blur	0.01
Distance	Sinkhorn

122

123 C Baselines

124 C.1 Real World Experiments

125 All baselines instantiate the encoder f_ϕ and decoder π_θ architecture described in Section B and
126 Table 1 with modifications to input and output spaces.

127 **Robot-only Behavioral Cloning (Robot-only BC).** This baseline is trained only on robot observa-
128 tions. The encoder includes vision stems for egocentric and wrist camera inputs and a proprioceptive
129 stem for joint-space inputs. The resulting tokens and M context tokens are passed to the transformer
130 trunk. The decoder head uses K learnable tokens to produce future actions, where each action is
131 a 7-dimensional Cartesian command: 3D position, 3D orientation, and scalar gripper. The loss is
132 computed using Smooth L1 for position and gripper, and MSE for orientation.

133 **Co-train.** Co-train uses both human and robot demonstrations during training and represents an
134 ablated version of the EgoBridge model, where the joint-OT loss ($\mathcal{L}_{OT-joint}$) is not applied.

135 **EgoMimic [4].** EgoMimic follows the Co-train architecture but includes two modifications: (1) The
136 egocentric RGB input is masked with an overlay and red line augmentation, following [4] before it is
137 input into the encoder, and (2) The decoder uses two heads: one for robot joint-space commands $\in R^7$
138 and another for shared cartesian pose $\in SE(3) + gripper$ for human and robot data. The joint-space

139 predictions are supervised using a Smooth L1 loss on the entire prediction vector, while cartesian
 140 pose predictions are supervised by the loss described in Section B. During training, the appropriate
 141 head is selected based on the data source, where robot data provides gradient updates through the
 142 joint predictions and cartesian predictions, while the human data provides gradient updates for the
 143 model through the cartesian predictions. The identical overlay is applied during evaluation.

144 **ATM [5].** For fair comparisons, we implement ATM with an identical architecture as our method
 145 and extend it with an additional head designed to predict future dense 2D motion trajectories in the
 146 image plane, as described in the original ATM paper. The goal is to test whether access to pixel-
 147 level dynamics improves downstream behavior cloning performance via improved latent structure.
 148 Ground-truth 2D pixel tracks are generated using CoTracker [6] and organized into a temporal stack
 149 of length 10, covering 100 future frames. The action-track head outputs future 36 keypoints across
 150 6×6 grid over the image plane. Training proceeds in two stages: first, the the trunk, stems, and the
 151 2D track head are pre-trained for 500 epochs to jointly predict human and robot 2D action tracks;
 152 second, the trunk and 2D head are frozen, and only the cartesian action head is fine-tuned on robot
 153 data. We opt for stacked, non-autoregressive, predictions of the full temporal keypoint trajectory,
 154 diverging from the sequential autoregressive formulation in the original ATM [5], in order to isolate
 155 the representational benefits of point-track conditioning on downstream imitation performance.

156 **MimicPlay [7].** MimicPlay is implemented as a hierarchical policy which consists of a "high-level"
 157 policy which is co-trained on both human and robot data and a "low-level" policy which is trained
 158 only on robot data and is conditioned on the context output of the "high-level" policy.

159 *High-level policy.* We parameterize the high level policy of MimicPlay with a *shared* vision stem
 160 for I_{ego} for both human and robot and a shallow "trunk" which consists where $N_{trunk} = 2$, while
 161 the other hyperparameters for the trunk remain the same to match the parameter size of the original
 162 high-level in MimicPlay. 2 tokens corresponding to the high level context are prepended to the vision
 163 tokens. The first 2 output tokens are the latent context z_{high} that are passed to the low-level policy. A
 164 KL-divergence loss is applied between the robot data z_{high}^R and the human data z_{high}^H . A Gaussian
 165 Mixture Model decoder with identical parameters to the MimicPlay paper, takes z_{high} and predicts
 166 the mixing coefficient, mean, and the variance of GMM. With these predictions, a sequence of 100
 167 3D positions are sampled from the GMM for a given input observation and supervised with the
 168 ground-truth actions for the observation with a negative log likelihood loss. The high-level planner is
 169 trained until convergence which is around 60000 iterations.

170 *Low-level policy.* The low level policy is parameterized the Robot-only BC model. The high-level
 171 policy is frozen and the latent output z_{high} is concatenated to the M tokens produced by f_ϕ which
 172 are then decoded into cartesian actions supervised by \mathcal{L}_{BC} on robot data.

173 C.2 Sim PushT

174 In the more controlled Sim PushT environment, we benchmark EgoBridge against standard domain
 175 adaptation techniques commonly used in feature-level alignment. All baselines instantiate the same
 176 encoder f_ϕ and decoder π_θ described in Section B.2, with differences only in the training objective
 177 and data source used.

178 **Maximum Mean Discrepancy [8].** We evaluate domain alignment using the MMD loss [8] as an
 179 alternative to the Joint OT loss. This baseline uses the same training configuration as EgoBridge but
 180 replaces the Joint-OT objective with the MMD loss applied on the latent embeddings from the shared
 181 encoder. The overall training objective is:

$$\mathcal{L} = \mathcal{L}_{BC-cotrain} + \lambda_{MMD} \cdot \mathcal{L}_{MMD}, \quad \text{where} \quad \lambda_{MMD} = 1.0$$

182 The MMD loss is computed as:

$$\mathcal{L}_{MMD} = \frac{1}{n^2} \sum_{i,j} k(z_H^{(i)}, z_H^{(j)}) + \frac{1}{m^2} \sum_{i,j} k(z_R^{(i)}, z_R^{(j)}) - \frac{2}{nm} \sum_{i,j} k(z_H^{(i)}, z_R^{(j)})$$

183 where $k(\cdot, \cdot)$ is a Gaussian RBF kernel:

$$k(x, y) = \exp \left(-\frac{\|x - y\|_2^2}{2\sigma^2} \right)$$

184 and σ is the kernel bandwidth hyperparameter.

185 **Standard OT.** This baseline implements marginal alignment using an unshaped Optimal Transport
 186 cost matrix without any DTW-based behavioural pairing. It follows the same formulation as Ego-
 187 Bridge (see Algorithm 1) but removes the DTW pairing step (Lines 11–17), using direct squared
 188 ℓ_2 distance $\tilde{C}_{ij} = D_{ij}$ for all (i, j) in the batch. This serves as an ablation that tests the benefit of
 189 behaviourally guided pairings in EgoBridge.

190 **Co-train.** Similar to the real-world setting, the co-train baselines ablates the Joint OT loss from the
 191 sim EgoBridge architecture to evaluate the baseline performance of training a shared policy on both
 192 embodiments.

193 **Target-only BC.** This baseline is trained using only robot demonstrations from the triangle pusher
 194 in the white background with original T configuration (100 trajectories total). It uses the same
 195 architecture as EgoBridge but is never exposed to data from the circle pusher (source domain).

196 D Robot Data

197 **Bimanual Manipulator.** To effectively utilize egocentric human data for manipulation, the robot
 198 hardware platform must resemble human sizes and kinematic workspaces. Drawing inspiration from
 199 the “Eve” robot platform introduced in EgoMimic [4], we develop a custom mobile manipulator
 200 that comprises of two 6-DoF ViperX 300s mounted in an identical inverted configuration on a
 201 height-adjustable rig. Similar to Eve, we propose to leverage the Project Aria glasses [9] as the
 202 main egocentric perception sensor for the robot and mount it in a way that emulates the hand-eye
 203 configuration of a human adult. This effectively mitigates the human-robot camera device gap and
 204 reduces the sensor-manipulator kinematic gap. Each arm is equipped with an Intel Realsense D405
 205 on its wrist to facilitate precise near-range manipulation. The raw RGB data from the aria glasses
 206 is undistorted to a linear camera model, resized and reshaped to $480 \times 640 \times 3$. Fig. 1 shows an
 207 annotated picture of the bimanual manipulator. The bimanual manipulator is teleoperated using a
 208 leader-follower system similar to ALOHA [10], where two WidowX 6-DoF arms in an inverted
 209 configuration as seen in Fig. 1

210 **Data Processing.** Each robot demonstration comprises synchronized streams of joint-space pro-
 211 prioception, RGB image observations, and action labels recorded at 50Hz. At each timestep t , we
 212 collect the current joint configuration $\mathbf{q}_t^R \in \mathbb{R}^{d_q}$, where d_q denotes the number of actuated joints
 213 for one or two robot arms. RGB images are recorded from two sources: (1) an egocentric camera
 214 ($I_{ego,t}^R \in \mathbb{R}^{480 \times 640 \times 3}$) mounted using Project Aria glasses in a head-like configuration, and (2)
 215 wrist-mounted RGB cameras ($I_{wrist,t}^R \in \mathbb{R}^{480 \times 640 \times 3}$) placed near the end-effectors.

216 The raw action at each timestep is defined as the commanded joint position $\mathbf{a}_t^R \in \mathbb{R}^{d_q}$, represent-
 217 ing the control input issued at time t . To obtain cartesian actions used for training, we use analytical
 218 forward kinematics to compute the $SE(3)$ cartesian pose. The cartesian pose is then projected into
 219 the image frame via the Aria glasses extrinsics (T_R^{aria}) obtained through hand-eye calibration, which
 220 represent the transformation between the robot base frame and the Aria glasses camera frame. The
 221 ground-truth actions $\mathbf{a}_t^R \in \mathbb{R}^{d_q}$ are obtained through

$$\mathbf{a}_t^R = (T_R^{aria})^{-1} \cdot \text{FK}(\mathbf{a}_{raw,t}^R)$$

222 E Embodied human data

223 We use the Project Aria [9] glasses to collect *human embodied experience data*. The glasses are
 224 accompanied by the Aria Machine Perception Services (MPS). The raw data from the Aria glasses
 225 consist of an RGB camera stream, SLAM cameras, IMU, eye cameras. This raw data is uploaded to a
 226 cloud service provided by Project Aria, known as the MPS. The service returns device pose estimated
 227 using the SLAM camera, hand tracking relative to the device frame, a semi-dense point-cloud of
 228 the environment and eye gaze estimation. This processed data is obtained as CSV files with aligned
 229 timestamps. The hand tracking is in the form of 3D positions $p^H \in SE(3) \times SE(3)$ for both hands
 230 in device frame, while the head pose $T^{aria} \in SE(3)$ in world frame. Each hand position p_t^H is in
 231 device frame T_t^{aria} for timestep t . The raw RGB data is undistorted to a linear camera model and
 232 reshaped to $480 \times 640 \times 3$.

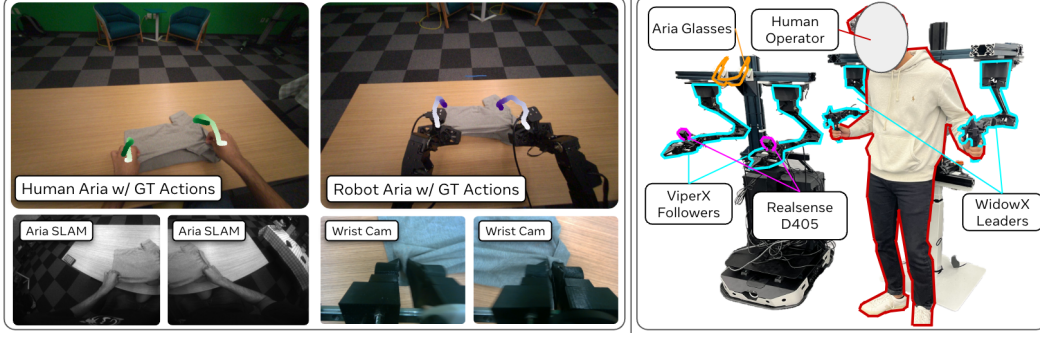


Figure 1: We employ the Aria glasses to capture Egocentric RGB images for both human and robot embodiments. The Aria uses its side SLAM cameras to estimate device pose and hand tracking (left). Ground truth (GT) cartesian actions for both embodiments are projected to pixel space using Aria intrinsics and visualized on the images. Our robot system is a leader-follower system with two Viper X arms as the followers and two Widow X arms as the leaders which are teleoperated by the human demonstrator (right).

Constructing human actions. One challenge with unifying the reference frames for joint policy learning is that robot actions are typically in a fixed reference frame (usually the base frame), while human hand tracking is in the device frame, which moves. Following the idea of predicting action chunks [3, 10], we aim to construct action chunks $a_{t:t+k}^H$. Taking the single-arm case without loss of generality, the raw trajectory is $SE(3)$ poses $[p_t^H, p_{t+1}^H, p_{t+2}^H, \dots, p_{t+k}^H]$, where each hand position p_t^H is in device frame T_t^{aria} . We choose to create *pseudo-reference frames* taking inspiration from [4], where we construct an action $a_{t:t+k}^H$ by projecting k future hand positions into the device frame at timestep t . This allows the policy to predict trajectories with respect to a stable reference at each timestep. As such, the trajectory is constructed by:

$$a_{t:t+k}^H = \left[(T_t^{aria})^{-1} T_{t+i}^{aria} \cdot p_{t+i}^H \right]_{i=1}^k$$

F Data processing and alignment.

Recent cross-embodiment works [4, 11, 12] show that individually normalizing proprioception and actions for both embodiments, helps co-training. Similarly, we employ *z-score* normalization by subtracting the per-embodiment dataset mean and dividing by the standard deviation

$$norm(p) = (p - \mu_p) / \sigma_p$$

We normalize actions identically, using per-dimension mean and standard deviation.

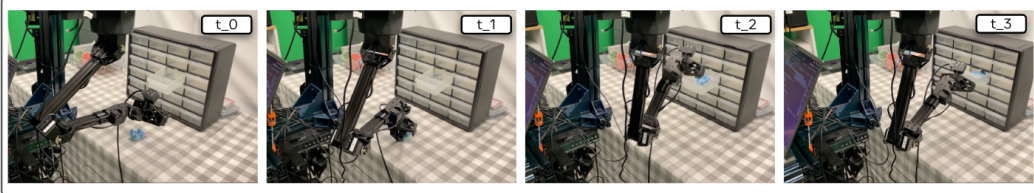
For both human and robot, our action chunk size $k = 100$. Human data sensor streams (RGB, hand tracking and SLAM) are received at 30hz. We construct a human action chunk by taking 10 sample pose values at an interval of 3 frames, and interpolating them to a trajectory of length 100. This corresponds to a trajectory of length 100 to 0.9 seconds in real time. We construct a robot action chunk by concatenating the ground-truth actions of 100 successive timesteps which correspond to 2 seconds in real time. The human and robot data sources are visualized in Fig. 1

G Additional Task Details

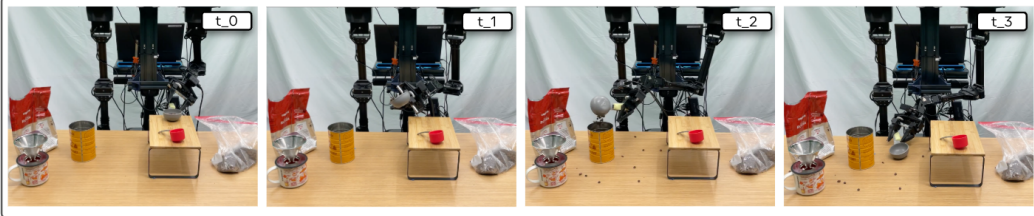
G.1 Real World Experiments

Drawer. For this task, we collect one hour of human data and 30 minutes of robot data comprising 144 demonstrations. The human data was equally distributed across all four quadrants, while the robot demonstrations were equally divided among three quadrants (excluding the top-right), with each receiving 48 robot demonstrations (8 demonstrations per drawer). The main failure modes for this task are the inability to correctly grasp the toy, taking the toy towards the incorrect drawer and

a) Drawer



b) Scoop Coffee



c) Laundry

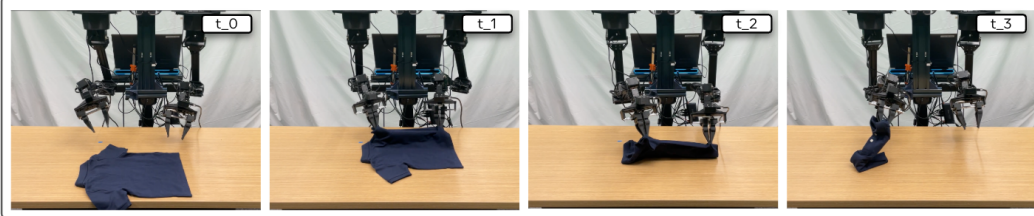


Figure 2: Qualitative success of EgoBridge on each of our real world tasks.

Table 4: Data collection overview for both Human(H) and Robot(R) data. We report both the number(#) of total task demonstrations and the time(min) took to collect them.

Task	H #	H min	H #/min	R #	R min	R #/min
Drawer	360	60	6	144	29	5
Scoop Coffee	720	180	4	50	13.3	3.75
Laundry	700	120	5.8	300	120	2.5

261 being unable to completely push the drawer shut.

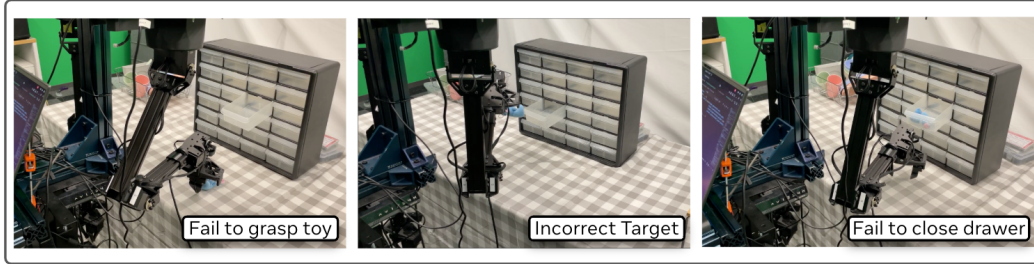
262 **Scoop Coffee.** The scoop coffee task emphasizes observation generalization with novel objects and
 263 scenes. To address this, we gathered 180 minutes of human data, equally split across three scenarios:
 264 the base scene, a new target object (grinder), and a new target object (grinder) with a new scene. For
 265 robot data, we collected 50 demonstrations with the original target object and scene configuration.
 266 The two primary failure modes in this task is an inability to precisely grasp the scoop and imprecise
 267 targeting of the target container.

268 **Laundry.** The laundry task demands precise bimanual coordination. We collected a similar amount
 269 of human data (2 hours) as the scoop task, comprising 700 demonstrations, alongside two hours of
 270 robot data (300 demonstrations). Common failure modes involve missing one or both sleeves during
 271 any of the three sub-stages, leading to an imprecise or unsuccessful fold. The qualitative task
 272 successes are visualized in Fig. 2 and common failure modes for each task are visualized in Fig. 3,
 273 while the data mixture is shown in Table 4.

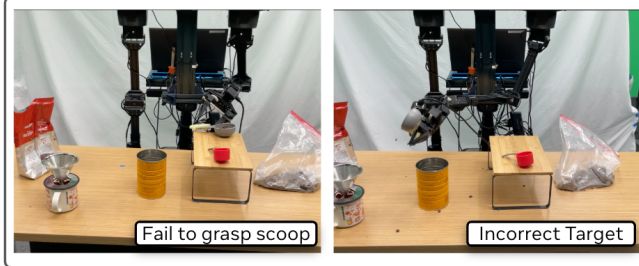
274 G.2 Sim PushT

275 We collect a total of 350 demonstrations across 4 scenarios using the PyMunk simulator environment
 276 published with Diffusion Policy [3]. We collect 100 demonstrations each for the circle pusher and
 277 triangle pusher in the original T configuration with the white background. In addition, we collect
 278 50 demonstrations each of the circle pusher with original T configuration on purple background,

a) Drawer - **Failure** Modes



b) Scoop Coffee - **Failure** Modes



c) Laundry - **Failure** Modes

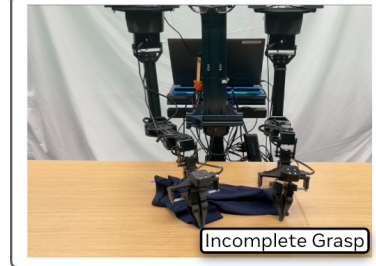


Figure 3: Common failure modes for each of the real world evaluation tasks.

279 mirrored T on purple background and mirrored T on white background. We apply a friction coefficient
280 multiplier of 0.7 on the triangle pusher to embody a kinematic gap like human and robot.

281 H Supplementary Simulation Results

282 We evaluate 3 settings : In distribution, which corresponds to the triangle pusher with the original T
283 configuration on a white background; Observation Generalization which corresponds to the previous
284 setting with a purple background; Observation and Behaviour Generalization which corresponds
285 to the triangle pusher with a mirrored T on a purple background. For evaluation, we select a total
286 of 100 seeds between 101 and 9999 using a deterministic random sampling with seed 42. The
287 complete results are summarized in Table 5. The Mean Reward is calculated using the average max
288 Intersection-over-Union with the goal across 100 seeds and Success Rate (SR) is computed using the
number of seeds with a reward of over 0.9.

Table 5: Sim PushT results across 3 evaluation settings

Method	In-distribution (Mean Reward SR)		Generalization			
			Purple Bg (Mean Reward SR)		Purple Bg + Mirrored T (Mean Reward SR)	
EgoBridge	0.7605	53.00%	0.7206	48.00%	0.6520	39.00%
Target Only	0.5555	39.00%	0.0904	0.00%	0.0992	0.00%
Cotrain	0.7062	48.00%	0.6899	42.00%	0.6214	31.00%
Standard OT	0.7009	38.00%	0.5303	15.00%	0.5109	8.00%
MMD	0.6439	45.00%	0.4867	22.00%	0.5876	14.00%

289

290 I Pseudo-pair Visualisation

291 To show why Dynamic Time Warping (DTW) is an ideal cost function to identify behaviourally
292 similar trajectories to align, we visualize randomly sampled mini-batch pairs from the dataset for
293 each of the tasks and qualitatively compare them with the Mean Square Error (MSE) cost function
294 pairs used for the MSE ablation. A pair, as defined earlier, is the row-wise cost function minimum in



Figure 4: Qualitative visualization of randomly sampled mini-batch pairing of MSE and DTW.

the cost matrix computed from a batch of human and robot data. Qualitatively, DTW is more robust to temporal shifts, viewpoint shifts and is more spatially precise when compared to MSE. Specifically, MSE often pairs the incorrect sub-task in tasks like Laundry, and loses precise spatial location of the target container in tasks like Drawer and Scoop Coffee. The few failure modes of DTW are where the trajectory overlaps a few different stages which leads to temporal misalignments and when locations are spatially apart but visually close due to an extreme viewpoint shift. Despite this, DTW enables pairings that are able to capture very fine-grained changes in trajectory to find the most behaviorally similar between human and robot data. This is visualized in Fig. 4

J Latent Space Visualisation

We visualize the learned latent feature outputs of f_ϕ for the EgoBridge and Co-train models using T-SNE for the Drawer and Scoop Coffee tasks. To evaluate EgoBridge’s ability to achieve joint distribution alignment, we use K-Nearest Neighbors to identify the closest learned human feature representations to a give robot feature representation. We plot the ground truth actions associated with those observations onto the input images and visualize the nearest neighbors. In addition to an overall lower Wasserstein-2 distance, which indicates global alignment, EgoBridge also aligns observations that correspond to similar behaviors in the latent space. This is visualized in Fig 5.



Figure 5: Qualitative visualization of the learned latent space using T-SNE and K-Nearest Neighbors.

References

- [1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.
- [2] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690, 2019.
- [3] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.
- [4] Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu. Egomimic: Scaling imitation learning via egocentric video, 2024.
- [5] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning, 2023.

- 325 [6] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and
326 Christian Rupprecht. Cotracker: It is better to track together, 2024.
- 327 [7] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and
328 Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play.
329 *arXiv preprint arXiv:2302.12422*, 2023.
- 330 [8] Arthur Gretton, Karsten Borgwardt, Malte J. Rasch, Bernhard Scholkopf, and Alexander J.
331 Smola. A kernel method for the two-sample problem, 2008.
- 332 [9] Jakob Engel, Kiran Somasundaram, Michael Goesele, Albert Sun, Alexander Gamino, Andrew
333 Turner, Arjang Talattof, Arnie Yuan, Bilal Souti, Brighid Meredith, Cheng Peng, Chris Sweeney,
334 Cole Wilson, Dan Barnes, Daniel DeTone, David Caruso, Derek Valleroy, Dinesh Ginjupalli,
335 Duncan Frost, Edward Miller, Elias Mueggler, Evgeniy Oleinik, Fan Zhang, Guruprasad Soma-
336 sundaram, Gustavo Solaira, Harry Lanaras, Henry Howard-Jenkins, Huixuan Tang, Hyo Jin Kim,
337 Jaime Rivera, Ji Luo, Jing Dong, Julian Straub, Kevin Bailey, Kevin Eickenhoff, Lingni Ma, Luis
338 Pesqueira, Mark Schwesinger, Maurizio Monge, Nan Yang, Nick Charron, Nikhil Raina, Omkar
339 Parkhi, Peter Borschowa, Pierre Moulon, Prince Gupta, Raul Mur-Artal, Robbie Pennington,
340 Sachin Kulkarni, Sagar Miglani, Santosh Gondi, Saransh Solanki, Sean Diener, Shangyi Cheng,
341 Simon Green, Steve Saarinen, Suvam Patra, Tassos Mourikis, Thomas Whelan, Tripti Singh,
342 Vasileios Balntas, Vijay Baiyya, Wilson Dreewes, Xiaqing Pan, Yang Lou, Yipu Zhao, Yusuf
343 Mansour, Yuyang Zou, Zhaoyang Lv, Zijian Wang, Mingfei Yan, Carl Ren, Renzo De Nardi,
344 and Richard Newcombe. Project aria: A new tool for egocentric multi-modal ai research, 2023.
- 345 [10] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual
346 manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.
- 347 [11] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep
348 Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yun-
349 liang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey
350 Levine. Octo: An open-source generalist robot policy, 2024.
- 351 [12] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo
352 Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow
353 model for general robot control, 2024. URL <https://arxiv.org/abs/2410.24164>, 2024.