SHEC3R: A Robot-Grounded Neural Initializer for Stationary Eye-to-Hand Calibration

Anonymous Submission

***Abstract*— Stationary eye-to-hand calibration, which estimates the transform between a fixed camera and the robot base, is a prerequisite for vision-guided manipulation. Marker-free methods based on mask-level differentiable rendering (DR) achieve high accuracy, but their silhouette objective is non-convex: they converge only from a close initial estimate and must solve a new optimization for every scene. The robot itself, however, constrains the camera–base transform through the image location of its base, the projection of its base axes, and the configuration of its arm. We present SHEC3R, which encodes these cues as robot-grounded prompts that condition a pretrained visual-geometry backbone. Given RGB frames and robot masks, SHEC3R predicts the base-to-camera transform and a dense point cloud in the robot base frame in a single forward pass, without a calibration board, robot meshes, or joint readings; one depth frame recovers the metric scale. When used to initialize the same DR stage, which does require meshes and joint readings, SHEC3R reaches a final rotation error below 5° on 80.2% of simulated and 86.0% of real scenes, compared with 8–25% for prior initializers. To train and evaluate SHEC3R, we introduce RoboEyeCalib, a benchmark with 2.83M synthetic frames from four simulated arms and 883 real scenes from five physical arms, and we additionally evaluate on the Flexiv Rizon 4 subset of RH20T. Code and data will be released.**

I. Introduction

Stationary hand-eye calibration (SHEC) [1], [2], also called eye-to-hand calibration, estimates the rigid transform between a fixed external camera and the robot base. The standard target-based procedure attaches a calibration board to the end effector and pairs its observations with the reported end-effector poses. Although the formulation is well established, it requires accurate forward kinematics, reliable target detection, and a data-collection routine that must be repeated whenever the camera is moved. Marker-free methods [3], [4], [5], [6], [7], [8], [9], [10] remove the board and instead rely on differentiable rendering, visual tracking, or geometric registration. Among them, mask-based differentiable rendering (DR) [4], [3] achieves high accuracy: it aligns rendered robot silhouettes with observed masks and, when it converges, recovers the extrinsic to within a fraction of a degree. However, the silhouette objective is highly non-convex. Its basin of attraction is narrow, and an optimization initialized outside this basin typically terminates in a local minimum far from the true pose.

The final accuracy of DR thus depends largely on its initialization, and existing initializers rarely place the estimate within the basin. For instance, when we fix the scenes and the DR stage and vary only the initializer (Section V-C), every initializer converges on 82–100% of the scenes in which it starts within 10° of the ground truth, yet prior initializers start within this range on only 1–17% of scenes. Therefore, rather than running a robot-specific optimization from a poor starting point at every deployment, we learn to predict an initial estimate inside the basin directly from RGB images and robot masks.

We introduce **SHEC3R**, which builds SHEC on a pre-trained visual-geometry foundation model (π^3 [11]). Given N RGB observations and per-frame robot masks, SHEC3R predicts both the base-to-camera transform $\hat{\mathbf{T}}_{cb}$ and a dense scene point cloud in the robot base frame in a single forward pass. The backbone is trained for multi-view reconstruction without robot-specific supervision: it receives no labels for the robot base frame or the arm configuration, and its predictions are expressed relative to the cameras, up to an unknown scale. SHEC3R therefore conditions it with input-dependent calibration prompts that encode three cues: the image location of the robot base, the projected directions of the base axes, which resolve the base orientation, and the arm configuration visible in the robot mask. Cross-frame attention over these prompts fuses evidence from multiple robot configurations into a single shared calibration.

The two outputs constrain each other: the extrinsic determines the frame in which the reconstruction is expressed, and the reconstructed geometry constrains where the base can lie. During training, we enforce this consistency by lifting the detected base location into 3D through the predicted point map and requiring it to agree with the predicted translation. At deployment, the prediction can be used directly as a feed-forward calibration that requires no robot model, or as the initialization of an optional DR stage, which additionally uses URDF meshes and joint readings.

For training and evaluation, we build **RoboEyeCalib**, a synthetic-plus-real calibration dataset spanning four simulated and five real robot arms. Unlike existing datasets [12], [13], [14], which target camera-to-robot pose estimation on one or a few robots or provide camera-to-base extrinsics only as a by-product of large-scale teleoperation on a single arm type, RoboEyeCalib is built for marker-free SHEC and combines large-scale synthetic supervision with real-world data across multiple robot morphologies.

In summary, our contributions are:

- We propose SHEC3R, a learned initializer for SHEC that conditions a pretrained visual-geometry backbone with robot-grounded calibration prompts and predicts $\mathbf{T}_{cb}$ and a base-frame point cloud without a calibration board, robot meshes, or joint readings.
- We introduce RoboEyeCalib, a synthetic-plus-real dataset

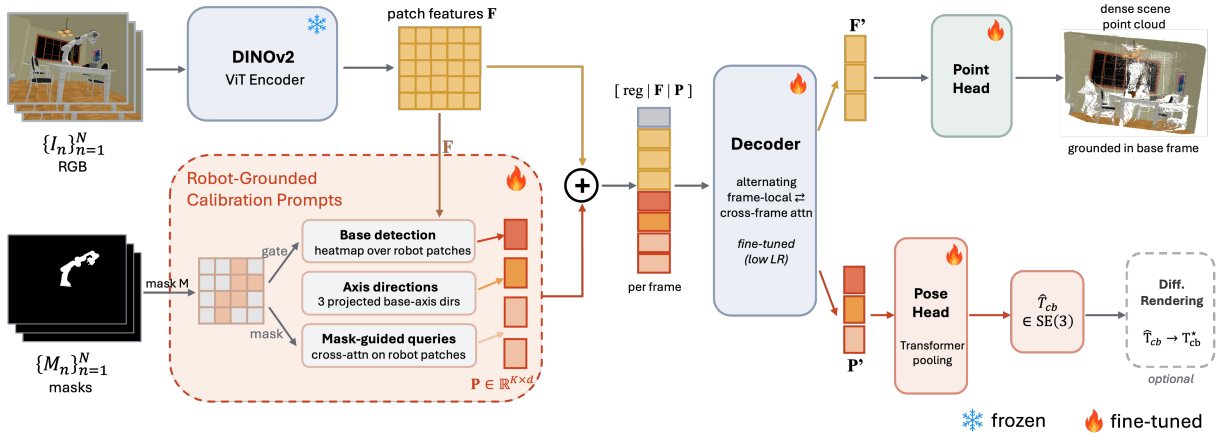


Fig. 1. **SHEC3R pipeline.** Given N RGB frames and robot masks, SHEC3R conditions a pretrained visual-geometry backbone with robot-grounded calibration prompts and predicts the base-to-camera transform $\mathbf{T}_{cb}$ and a scene point cloud in the robot base frame. The prediction is used directly or as the initialization of an optional differentiable-rendering (DR) stage.

and benchmark for marker-free SHEC that covers diverse cameras, scenes, and robot morphologies, with camera–base layouts held out for evaluation.

- We show that SHEC3R provides a reliable initialization for DR refinement, improving calibration accuracy and robustness over prior marker-free methods on simulated and real robots, including the external RH20T platform.

II. Related Work

Classical stationary hand-eye calibration. Classical methods [1], [2] estimate the camera-to-base transform from paired robot and target observations, assuming accurate robot kinematics, a calibration target, and carefully collected motion sequences. These assumptions make the problem mathematically clean but are inconvenient at deployment scale, since every new robot–camera setup repeats the procedure.

Marker-free and optimization-based calibration. Recent methods replace the board with robot-centric visual cues and solve a per-scene optimization [4], [3], [9], [6], [8]. EasyHeC [4] aligns rendered and observed robot masks by differentiable rendering from a keypoint-based initialization, and EasyHeC++ [3] replaces this initialization with a sampling-based search; Kalib [9] tracks reference points and solves PnP, and Hydra [6] registers the robot model to RGB-D observations. These methods require the robot kinematic model, per-frame joint readings, or both, and their accuracy depends on how close the optimization starts to the solution. We quantify this dependence in Section V-C, where SHEC3R supplies the initialization for EasyHeC's mask-based DR stage.

Camera-to-robot pose estimators address a related problem. RoboPose [15] estimates the robot pose and joint angles by render-and-compare against the robot CAD model, whereas CtRNet-X [16] detects 2D keypoints and solves PnP against 3D keypoints obtained from joint angles through forward kinematics, using differentiable rendering only for self-supervised training. Both therefore require the robot model at inference.

Robotics vision foundation models. Visual-geometry foundation models learn reusable priors for camera pose, depth, point clouds, and multi-view structure [17], [18], [19], [20], [21], [22], [11], [23], [24]. Recent work applies these priors to robot calibration, but for cameras moving with the robot. Bi-JCR [7] calibrates wrist-mounted cameras on two arms by combining DUSt3R [17] reconstructions with forward kinematics, and Calib3R [5] jointly optimizes MASt3R [18] pointmaps and robot poses for cameras mounted on manipulators or mobile robots. Robo3R [10] conditions a visual-geometry backbone on robot state to predict metric reconstruction in the robot frame. SHEC3R addresses the stationary eye-to-hand setting: it requires neither joint angles nor predefined per-robot keypoints at inference, and recovers geometry in the robot base frame from static observations.

Datasets and benchmarks. Existing datasets do not support training a marker-free stationary calibration model across robots. Pose-estimation datasets such as DREAM [13] and the keypoint dataset of Lu et al. [12] cover one or a few robot types. Large-scale manipulation datasets offer diverse scenes but limited calibration data: DROID [14] contains only Franka arms, which our data already cover, with checkerboard extrinsics later refined automatically for a subset of episodes; Open X-Embodiment [25] aggregates datasets whose camera modalities vary by subset, with depth in only some of them, and does not document camera-to-base extrinsics; and RoboMIND [26] records four embodiments with RGB-D cameras but likewise does not describe camera-to-base extrinsics. Among the datasets we examined, only RH20T [27] provides scene-varying camera-to-base extrinsics, RGB-D observations, and joint states on an arm outside our own data. RoboEyeCalib (Section IV) addresses this gap with synthetic and real data across multiple arms and held-out camera–base layouts, and uses the Flexiv Rizon 4 subset of RH20T for external evaluation.

III. Method

We formulate stationary hand-eye calibration (HEC) as robot-grounded visual geometry. Given N RGB frames

$\{I_n\}_{n=1}^N$ captured by a fixed external camera and corresponding robot masks $\{M_n\}_{n=1}^N$, our goal is to estimate the rigid transform $\mathbf{T}_{cb} \in \mathrm{SE}(3)$ from the robot base frame to the camera frame, and to predict dense scene geometry expressed in the robot base frame. Because camera and base are both fixed, $\mathbf{T}_{cb}$ is shared across all frames, while different robot configurations supply complementary visual evidence. Figure 1 gives an overview.

Visual-geometry backbone. SHEC3R builds on a pretrained feed-forward 3D reconstruction backbone [11]. Each frame is encoded into patch tokens by a DINOv2 ViT encoder [28], followed by a multi-view decoder with alternating frame-local and cross-frame attention that produces per-frame dense point maps and camera-pose features. We freeze the encoder and fine-tune the decoder at a low learning rate, preserving the pretrained visual-geometry prior while adapting it to robot-frame grounding. This suits stationary calibration because estimating $\mathbf{T}_{cb}$ is not independent of reconstruction: the extrinsic defines the frame in which scene geometry becomes actionable, while reconstructed geometry provides the spatial evidence for the base pose. SHEC3R therefore does not replace reconstruction with pose regression; it queries the reconstruction prior with robot-specific cues.

Robot-grounded calibration prompts. We convert encoder features and robot masks into K input-dependent prompt tokens. Let $F \in \mathbb{R}^{hw \times d}$ denote the patch features of one frame and let M be the robot mask pooled to the patch grid. *Base-location prompt.* A lightweight patch-wise detector predicts a heatmap over robot-mask patches. The resulting attention weights select the feature most likely to correspond to the robot base origin, producing a base token that acts as an explicit spatial anchor for the calibration problem.

Axis-orientation prompt. Localizing the base origin does not determine the full 6D transform. We additionally predict the projected 2D directions of the three base-frame axes near the detected base region and aggregate them into an axis token. This supplies an image-space orientation cue that helps resolve the rotational symmetries common in robot bases. The base frame's origin and axis directions are a software convention with no intrinsic visual definition, so this branch is supervised explicitly rather than left to emerge from the pose loss.

Mask-guided arm prompts. The remaining tokens summarize the visible robot configuration. Rather than average-pooling masked features, a set of learnable queries cross-attends to robot-mask patches, retaining spatial information about link orientation and end-effector direction as further evidence for the shared base pose.

The prompt set $P = [\, P_{\mathrm{base}}, P_{\mathrm{axis}}, P_{\mathrm{arm}} \,] \in \mathbb{R}^{K \times d}$ is appended to the decoder token sequence of each frame. Frame-local attention lets the prompts interact with that frame's geometry; cross-frame attention fuses prompt evidence from all robot configurations into a single stationary estimate.

Base-to-camera pose head. We collect the prompt states from all N frames and aggregate them with a small Transformer pooling head. The pooled representation maps to a 6D rotation representation [29] and a 3D translation, and the rotation is orthogonalized into a valid rotation matrix, yielding $\hat{\mathbf{T}}_{cb} \in \mathrm{SE}(3)$. Since the camera is stationary we do not rely on relative camera-pose tokens, whose ground-truth motion is degenerate in this setting.

Robot-frame point map prediction. In parallel, the decoder outputs a dense per-frame point map in the camera frame, transformed into the base frame by the estimated calibration, $\hat{\mathbf{X}}_b = \hat{\mathbf{T}}_{cb}^{-1} \hat{\mathbf{X}}_c$. This yields a robot-grounded point cloud as an additional output and provides the 3D evidence used by the detection-anchored losses below.

Detection-anchored training objective. Regressing $\mathbf{T}_{cb}$ alone does not force the detected base, the reconstructed point cloud, and the predicted extrinsic to be mutually consistent. We therefore anchor all three to a common 3D base point. The base detector is supervised by a per-patch binary cross-entropy against a Gaussian heatmap, with a standard deviation of one patch, centered at the projected ground-truth base origin. The axis branch is supervised by a cosine loss on the projected base-axis directions, weighted by the same heatmap and restricted to axes whose projected 5 cm segment is at least one pixel long.

Because the camera and the base are fixed, all frames share one camera frame. Let $\hat{\mathbf{x}}_n(p)$ denote the point of $\hat{\mathbf{X}}_c$ at pixel p of frame n, and let $\hat{\mathbf{t}}$ denote the translation of $\hat{\mathbf{T}}_{cb}$, both expressed in this frame. We divide both by $s = \frac{1}{|V|} \sum_{n,p \in V} \|\hat{\mathbf{x}}_n(p)\|_2$, where V is the set of pixels with valid ground-truth depth, which makes the following losses invariant to the global scale of the reconstruction. For frame n, let $\bar{\mathbf{x}}_{n,j}$ be the point map averaged over patch j, and let $\alpha_{n,j}$ be the softmax weight of the base detector with temperature 0.1, computed over the patches that are at least 30% covered by the robot mask. The lifted base point is $\mathbf{b}_n = \sum_j \alpha_{n,j} \bar{\mathbf{x}}_{n,j}$, and we define

$$
\begin{aligned}
\mathcal{L}_{\mathrm{lift}} &= \frac{1}{3N} \sum_{n=1}^{N} \sum_{c} h_{0.05}\big(b_{n,c} - \hat{t}_c\big), \\
\mathcal{L}_{\mathrm{xf}} &= \frac{1}{3} \sum_{c} \Big(\frac{1}{N} \sum_{n=1}^{N} \big(b_{n,c} - \bar{b}_c\big)^2 + \epsilon \Big)^{1/2}, \\
\mathcal{L}_{\mathrm{sc}} &= \frac{1}{|\Omega|} \sum_{p \in \Omega} \Big(\frac{1}{N} \sum_{n=1}^{N} \big\| \hat{\mathbf{x}}_n(p) - \boldsymbol{\mu}(p) \big\|_2^2 + \epsilon \Big)^{1/2},
\end{aligned}
\tag{1}
$$

where $c \in \{x, y, z\}$ indexes coordinates, h_δ is the Huber loss with threshold δ, $\bar{\mathbf{b}}$ and $\boldsymbol{\mu}(p)$ are averages over frames, $\epsilon = 10^{-6}$, and Ω is the set of pixels that have valid depth and lie outside the robot mask in all N frames. $\mathcal{L}_{\mathrm{lift}}$ ties the detection, the reconstruction, and the predicted translation to a single 3D point; $\mathcal{L}_{\mathrm{xf}}$ requires all frames to agree on this point; and $\mathcal{L}_{\mathrm{sc}}$ keeps static scene points consistent across frames, which prevents cross-frame attention from collapsing into independent single-frame reconstruction. The total objective is

$$
\begin{aligned}
\mathcal{L} = \mathcal{L}_{\mathrm{point}} + \lambda_{cb}\mathcal{L}_{cb} + \lambda_{\mathrm{det}}\mathcal{L}_{\mathrm{det}} + \lambda_{\mathrm{axis}}\mathcal{L}_{\mathrm{axis}} \\
+ \lambda_{\mathrm{lift}}\mathcal{L}_{\mathrm{lift}} + \lambda_{\mathrm{xf}}\mathcal{L}_{\mathrm{xf}} + \lambda_{\mathrm{sc}}\mathcal{L}_{\mathrm{sc}},
\end{aligned}
\tag{2}
$$

where $\mathcal{L}_{\mathrm{point}}$ is the scale-aligned point-map loss of the backbone, and $\mathcal{L}_{cb}$ combines a Huber loss ($\delta = 0.1$) on the

translation, rescaled by the same alignment factor as $\mathcal{L}_{\text{point}}$, a translation-direction cosine loss, and geodesic and column-cosine rotation losses. We set $\lambda_{cb} = 0.2$, $\lambda_{\text{det}} = 5$, $\lambda_{\text{axis}} = 1$, $\lambda_{\text{lift}} = 0.5$, $\lambda_{\text{xf}} = 0.2$, and $\lambda_{\text{sc}} = 0.5$.

We train in two stages: synthetic pretraining with exact masks, camera parameters, and base-to-camera labels, followed by real-world fine-tuning that handles sensor noise, lighting variation, segmentation error, and hardware-specific appearance. Stage 1 runs for up to 30 epochs of 5,000 iterations on eight NVIDIA GeForce RTX 2080 Ti GPUs (about 3.5 days), and we use the epoch-26 checkpoint selected on the validation split. Stage 2 fine-tunes this checkpoint for 15 epochs of 200 iterations on four GPUs of the same type. Both stages use AdamW with a one-cycle cosine schedule; the learning rates of the new modules and of the decoder are 10^{-4} and 10^{-5} in Stage 1 and are reduced by $10\times$ in Stage 2. Unless stated otherwise we use $K = 4$ prompt tokens: one base, one axis, and two mask-guided arm tokens.

Differentiable-rendering refinement. The feed-forward prediction can be used directly or refined by a local differentiable-rendering stage [30]. Since monocular reconstruction is up to scale, we first recover a metric scale from one RGB-D frame as the median ratio of measured to predicted depth over robot-mask pixels, and apply it to the translation and the point cloud. Starting from $\hat{\mathbf{T}}_{cb}$ we then optimize a small pose update $\Delta\boldsymbol{\xi}$, applied on the left of the initial estimate and shared by all N frames, that minimizes the discrepancy $\mathcal{E}$ between rendered and observed robot masks:

$$\begin{aligned} \Delta\boldsymbol{\xi}^{\star} &= \arg\min_{\Delta\boldsymbol{\xi}\in\mathbb{R}^6} \mathcal{E}\big(\exp(\Delta\boldsymbol{\xi}^{\wedge})\hat{\mathbf{T}}_{cb}\big), \\ \mathbf{T}_{cb}^{\star} &= \exp\big((\Delta\boldsymbol{\xi}^{\star})^{\wedge}\big)\hat{\mathbf{T}}_{cb}. \end{aligned} \tag{3}$$

Following EasyHeC [4], we solve for a local minimizer with Adam starting from $\Delta\boldsymbol{\xi} = \mathbf{0}$. This refinement is optional: the learned model provides a fast global estimate, while mask-based optimization supplies local geometric precision when needed. In contrast to optimization-only calibration, the renderer is initialized near the correct solution, which improves convergence and robustness (Section V-C). Unlike the feed-forward stage, however, rendering requires the URDF link meshes and per-frame joint readings mapped through forward kinematics, so we report the refined results separately from the robot-model-free ones.

IV. RoboEyeCalib Dataset

Overview. RoboEyeCalib benchmarks robot-grounded stationary hand-eye calibration across diverse robots, viewpoints, and scenes. It pairs large-scale synthetic data, which carries exact labels, with real deployments, which carry sensor noise, lighting variation, segmentation error, and hardware-specific appearance. It covers Franka Panda, xArm6, SO-100, and WidowX AI in simulation, and Franka (Panda and Research 3), AIRBOT Play, SO-101, OpenManipulatorX, and UR5e in the real world. We additionally convert the Flexiv Rizon 4 portion of RH20T [27] into the same format as an external sixth real platform. Figure 2 shows samples and Table I summarizes the splits.

TABLE I

RoboEyeCalib summary. A *scene* is one camera–base configuration with N=6 frames. Synthetic data follow an approximate 8:1:1 split. RH20T [27] is evaluated but not released. The 88 real test scenes include the 44-scene four-arm main split.

Source	Split	Arms	Scenes ($\mathbf{T}_{cb}$)	Frames
Synthetic (ours)	train+val	4	425,208	2,551,248
	test	4	46,739	280,434
Real (ours)	train+val	5	795	4,770
	test	5	88	528
RH20T [27] (external)	train+val	1	11,305	67,830
	test	1	561	3,366

Synthetic data. We generate synthetic data in ManiSkill3 [31] using tabletop scenes (Table, KitchenCounter), RoboCasa [32] kitchen layouts, ReplicaCAD [33], [34] room-scale apartments, and AI2-THOR [35] environments (ProcTHOR [36], RoboTHOR [37], iTHOR, ArchitecTHOR). To match the stationary hand-eye setting, each episode follows a fixed-view, varying-pose protocol: camera intrinsics and extrinsics and robot base placement are fixed, while the arm is articulated into six randomly sampled valid joint configurations. Each frame records RGB, depth, robot segmentation masks, and joint states (*qpos*); the simulator provides exact $\mathbf{T}_{cb}$ labels. Across four arms, eight environment families, and 8,056 background scene instances this yields 471,947 camera–base configurations and 2.83M labelled frames. For each arm, the training, validation, and test splits use disjoint background scene instances.

Real-world data. Our real split contains 883 scenes (5,298 frames, N=6 per scene) across five physical arms, collected in several laboratories: Franka (494 scenes, combining the Franka Emika Panda and its successor, the Franka Research 3 (FR3), which share the same kinematic structure and are reported as one arm), AIRBOT Play (114), SO-101 (110), OpenManipulatorX (106), and UR5e (59). We hold out 88 scenes for testing, *with no camera placement shared across splits* for the main four-arm comparison. This is the held-out variable throughout the paper: new camera–base layouts and scenes, not new robot morphologies.

Cameras. Real-world data are captured with 15 fixed external camera units at three image resolutions, covering both active and passive stereo sensing: Intel RealSense D415 and D435 (active IR stereo) and Stereolabs ZED 2i (passive stereo). The Franka data are the most diverse, spanning five sites and both sensor types. Only fixed external cameras are used for calibration, and RH20T contributes seven additional calibrated RealSense cameras.

Robot masks. Synthetic masks are provided by the simulator. Real-world masks are generated automatically by a Grounding DINO [38] + SAM 2 [39] pipeline.

Ground-truth calibration. Reference $\mathbf{T}_{cb}$ values are obtained once per physical setup using a two-stage board-based protocol. We first establish the rigid transform $\mathbf{T}_{bt}$ from a ChArUco board to the robot base using known board placement. We then estimate the board-to-camera transform

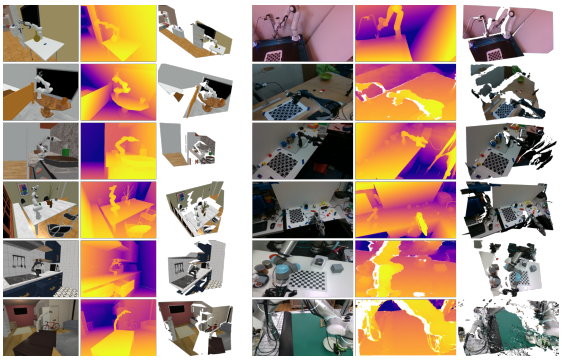


Fig. 2. **RoboEyeCalib overview.** RGB, depth, and back-projected point clouds for synthetic scenes across four arms and eight environment families (left), and for real-world setups of six physical arms captured with Intel RealSense and Stereolabs ZED cameras (right).

TABLE II

SYNTHETIC RESULTS PER ROBOT (100 SCENES PER ARM; MEDIANS, e_R IN °, e_t IN CM). *Stage*: + DR IS REFINED; PNP AND REG. ARE THE KALIB AND HYDRA PIPELINES. **BOLD**: BEST.

		Franka Panda		**SO-100**		**xArm6**		**WidowX AI**	
Method	Stage	e_R	e_t	e_R	e_t	e_R	e_t	e_R	e_t
EasyHeC	+ DR	99.69	54.12	0.56	0.59	102.43	25.98	102.66	31.25
EasyHeC++	+ DR	82.51	35.86	88.57	43.65	82.97	26.83	100.68	53.54
Kalib	PnP	53.38	120.44	13.97	14.62	57.76	70.40	29.83	28.84
Hydra	reg.	1.88	9.55	0.26	0.20	1.46	7.31	0.24	0.26
SHEC3R	+ DR	**0.05**	**0.07**	**0.05**	**0.05**	**0.08**	**0.14**	**0.04**	**0.04**

$\mathbf{T}_{ct}$ from detected corners via PnP and compute $\mathbf{T}_{cb} = \mathbf{T}_{ct}\mathbf{T}_{bt}^{-1}$. These references are used for both real-domain fine-tuning and evaluation.

V. EXPERIMENTS

Our experiments ask whether a learned initialization makes DR refinement converge reliably, and how far it transfers beyond the robots and cameras it was trained on.

A. Protocol and Baselines

We compare against four marker-free methods: EasyHeC [4], EasyHeC++ [3], Kalib [9], and Hydra [6], which represent segmentation-and-rendering, improved-initialization-and-rendering, tracking-and-PnP, and RGB-D registration pipelines, respectively. All methods are evaluated with the same camera intrinsics, $N{=}6$ observations, and train/test splits. Baselines are given the inputs required by their original pipelines, including masks, joint states, URDF meshes, or RGB-D when applicable. The feed-forward network of SHEC3R uses RGB images and robot masks and predicts rotation and translation up to scale; one depth frame recovers the metric translation scale. For EasyHeC, EasyHeC++, and SHEC3R, we apply the same multi-frame DR refinement to the initial estimate; Kalib and Hydra use their own PnP and registration pipelines. We report geodesic rotation error e_R in degrees and translation error e_t in centimeters, using medians over test scenes. Kalib succeeds on fewer than 70% of both synthetic and real scenes under our discrete wide-baseline protocol, so we report its errors only on successful cases.

B. Main Comparisons

We evaluate on four simulated arms with 100 test scenes each and on six real arms, including the Flexiv Rizon 4 from RH20T (Section IV), and report results per robot.

Table II reports results on the four simulated arms. SHEC3R achieves the lowest median rotation and translation errors on all four arms, with rotation errors of 0.04–0.08° compared with 0.24–1.88° for Hydra, the strongest baseline. EasyHeC and EasyHeC++ often fail to initialize DR inside the correct basin (Section V-C), and Kalib's estimates carry large rotation errors on all four arms. Two medians are optimistic: EasyHeC solves more than half of the SO-100 scenes, so its 0.56° hides a 41.23° mean, and Kalib succeeds on only 62% of WidowX AI scenes. Figure 4 shows qualitative comparisons.

Table III reports results on the six real arms, listing the model-free feed-forward mode of SHEC3R separately from the methods that use a robot model. The test sets contain 12 scenes for Franka and AIRBOT Play, 10 for OpenManipulatorX and SO-101, 7 for UR5e, and 561 for Rizon 4. With DR refinement, SHEC3R achieves the lowest median rotation error on Franka, AIRBOT Play, UR5e, and Rizon 4, and the lowest median translation error on these four arms and OpenManipulatorX. On SO-101, Hydra achieves the lowest errors on both metrics. Kalib solves only 8–43% of the scenes per arm, and its median rotation error on the solved scenes exceeds 29° on every arm. The largest margins appear on UR5e and Rizon 4, where all baselines have median rotation errors of 60–134°, whereas SHEC3R reaches 0.96° and 0.04°. On Rizon 4, SHEC3R also attains a median refined mask IoU of 0.996, compared with 0.26–0.37 for EasyHeC++ and Hydra. Figure 5 shows qualitative comparisons on real-world scenes.

C. Initialization Quality Controls the Basin of Attraction

To test whether initialization determines the success of refinement, we fix the scenes and the DR stage and vary only the initializer. EasyHeC, EasyHeC++, and SHEC3R provide initial estimates to the identical multi-frame DR refinement. We evaluate on 400 simulated scenes (100 each for Franka Panda, SO-100, xArm6-Robotiq, and WidowX AI), 86 real scenes from our self-collected arms, and 561 RH20T Rizon 4 scenes, computing metrics on the scenes common to the compared methods. EasyHeC is evaluated only in simulation because it requires a per-robot keypoint detector.

Under the shared refinement (Table IV), SHEC3R reaches an S@5° of 80.2% in simulation and 86.0% on our real scenes, compared with 25.2% for EasyHeC and 8.0–15.1% for EasyHeC++. Including the RH20T Rizon 4 scenes raises the S@5° of SHEC3R to 90.3% and lowers that of EasyHeC++ to 7.9%.

Table V groups the scenes by initial rotation error. When the initial error is below 10°, all methods succeed on 82–100% of scenes; beyond 45°, all methods succeed on at most 8%. The basin of the shared refinement bounds what can succeed, while the initializer determines which bin a scene starts in. The methods therefore differ mainly in the

TABLE III

REAL-WORLD RESULTS PER ROBOT (MEDIANS, e_R IN °, e_t IN CM). FF SHEC3R USES NO ROBOT MODEL; ALL OTHER COLUMNS DO. **BOLD**: BEST AMONG ROBOT-MODEL COLUMNS. KALIB: SOLVED SCENES ONLY. †PANDA AND FR3. ¶FROM RH20T [27], FINE-TUNED ON ITS TRAINING SPLIT.

		SHEC3R FF		**SHEC3R** +DR		EasyHeC++		Kalib			Hydra	
Robot	N	e_R	e_t	e_R	e_t	e_R	e_t	e_R	e_t	solved	e_R	e_t
Franka†	12	14.64	12.16	**0.68**	**2.16**	32.58	38.42	29.80	446.83	1/12	5.15	7.37
AIRBOT Play	12	5.36	6.91	**1.58**	**2.73**	66.87	62.35	48.77	310.33	4/12	2.74	2.78
OpenManipulatorX	10	5.75	7.73	1.64	**1.07**	**1.49**	1.17	68.01	55.52	2/10	6.03	2.58
SO-101	10	7.64	18.55	3.85	2.13	88.13	22.20	79.46	38.24	4/10	**1.99**	**1.59**
UR5e	7	8.73	41.15	**0.96**	**0.99**	73.07	49.89	80.1	88.6	3/7	96.76	24.02
Rizon 4¶	561	0.41	8.37	**0.04**	**0.05**	133.72	60.69	77.43	72.6	148/561	59.57	53.72

TABLE IV

SAME SCENES AND REFINEMENT, DIFFERENT INITIALIZERS. MEDIANS. *Real (all)* ADDS 561 RH20T RIZON 4 SCENES. S@k: PERCENTAGE OF SCENES WITH FINAL ERROR BELOW k; INIT IN BASIN: PERCENTAGE WITH INITIAL ROTATION ERROR BELOW 10°.

	Method	Init rot°	Final rot°	Final trans cm	S@2°	S@5°	S@5° & 2cm	Init in basin
Sim (400)	EasyHeC	80.2	80.5	26.3	23.8	25.2	23.8	16.5
	EasyHeC++	90.2	88.6	41.3	7.8	8.0	7.8	1.0
	SHEC3R	**5.8**	**0.05**	**0.07**	**78.8**	**80.2**	**77.8**	**61.5**
Real (86)	EasyHeC++	78.9	80.7	49.0	15.1	15.1	9.3	9.3
	SHEC3R	**5.9**	**0.74**	**1.04**	**75.6**	**86.0**	**59.3**	**70.9**
Real (all) (647)	EasyHeC++	129.4	128.2	60.1	7.7	7.9	7.0	3.6
	SHEC3R	**0.46**	**0.04**	**0.05**	**86.1**	**90.3**	**81.6**	**96.1**

TABLE V

SUCCESS RATE BY INITIAL ROTATION ERROR. S@5° (%) AFTER THE SHARED REFINEMENT; SUBSCRIPTS GIVE THE NUMBER OF SCENES PER BIN, AND "–" MARKS EMPTY BINS.

	Method	$[0,5)°$	$[5,10)°$	$[10,20)°$	$[20,45)°$	$[45,180)°$
Sim (400)	EasyHeC	96_{28}	97_{38}	67_{21}	31_{62}	2_{251}
	EasyHeC++	100_{4}	–	54_{13}	29_{73}	0_{310}
	SHEC3R	99_{179}	99_{67}	89_{47}	58_{55}	8_{52}
Real (86)	EasyHeC++	100_{7}	100_{1}	67_{3}	20_{15}	0_{60}
	SHEC3R	92_{39}	82_{22}	86_{22}	33_{3}	–
Real (all) (647)	EasyHeC++	100_{21}	100_{2}	43_{7}	36_{67}	0_{550}
	SHEC3R	91_{600}	82_{22}	86_{22}	33_{3}	–

fraction of scenes initialized inside the basin (last column of Table IV): SHEC3R places 61.5% (sim) and 70.9% (real) of scenes within 10°, compared with 16.5% for EasyHeC and 1.0% (sim) and 9.3% (real) for EasyHeC++.

D. Generalization

Adapting to a new robot. Rizon 4 appears in neither synthetic pretraining nor our self-collected real data. We evaluate on its 561 RH20T test scenes both the model fine-tuned on our five self-collected arms and a model fine-tuned on the RH20T Rizon 4 training split. Without Rizon 4 training data, SHEC3R reaches median rotation errors of 67.6° feed-forward and 71.3° after refinement, as its initialization lies outside the basin of the DR stage. After fine-tuning on Rizon 4 data, these errors drop to 0.41° and 0.04°, respectively, matching Table III. Extending SHEC3R to a new robot requires training data from that robot.

Training-data composition. We study the effect of synthetic pretraining on the 44 real test scenes of the four-arm main split (Franka, AIRBOT Play, OpenManipulatorX, and SO-101). Figure 3 compares real-world fine-tuning with and without synthetic pretraining under the same schedule. Synthetic pretraining lowers both the feed-forward and refined errors at every evaluated epoch, and the gap remains at the final epoch, where the mean refined rotation error is about 5° compared with 8° without pretraining. Synthetic pretraining alone, however, transfers poorly to real images, with rotation errors of 25–103° across the four real arms, so both data sources are needed.

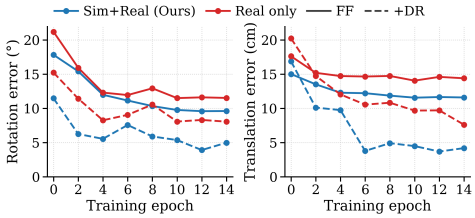


Fig. 3. **Effect of synthetic pretraining.** Mean errors during real-world fine-tuning on the 44 scenes of the four-arm main split.

Unseen camera intrinsics. SHEC3R does not take the camera intrinsics as input, so we test whether it generalizes beyond the fields of view (FoV) seen in training. We render 32 scenes at 9 vertical FoVs with the scene, robot state, and camera pose fixed, train on FoVs of 45–75°, and hold out 30–40° and 80–90°. Across this 3.7× range of focal lengths, the median rotation error stays within 2.9–7.7° and the translation error within 6.8–10.5 cm, without an abrupt increase at the boundaries of the training range. Supplying the intrinsics explicitly remains a natural extension (Section VI).

E. Base Visibility

In many eye-to-hand installations, the camera mainly observes the workspace and the upper arm, so the robot base may be only partly visible or outside the image. Since every scene in the main real-world test split contains the base, we evaluate this setting on a Franka subset of RoboEyeCalib in which the base is partly or fully out of frame, with 90 scenes for fine-tuning and 25 for testing. As base visibility decreases, the feed-forward rotation error of SHEC3R increases from 3.7° with the base in frame (15 scenes) to 11.0° at the image edge (3 scenes) and 17.6° out of frame (7 scenes), while the translation error stays within 7.6–10.8 cm without a consistent trend. The visible arm still constrains the robot position, whereas the base is needed to resolve its orientation.

The base-location and axis prompts are computed only

TABLE VI

ARCHITECTURE ABLATION

(REAL; MEANS, °/CM).

Variant	FF		+DR	
	e_R	e_t	e_R	e_t
Full	10.28	11.04	4.87	3.47
w/o axis	12.70	13.33	6.98	8.48
w/o arm	8.14	10.93	3.68	4.77
pool. + SE(3)	8.47	11.41	3.95	4.26

TABLE VII

CONSISTENCY-LOSS ABLATION

(SIM; MEANS, CM).

Variant	lift residual	cross-frame spread	scene incons.
Full	7.52	1.75	1.08
w/o $\mathcal{L}_{\text{lift}}$	**12.31**	1.71	1.02
w/o $\mathcal{L}_{\text{xf}}$	9.38	**3.87**	1.04
w/o $\mathcal{L}_{\text{sc}}$	5.83	0.89	**1.36**

over visible robot-mask patches, whereas the mask-guided arm prompts are always active. When the base projects outside the image, the base detector is supervised with an all-zero target and does not predict a base location. SHEC3R then relies on the arm prompts alone, consistent with the larger degradation in rotation than in translation.

F. Ablations

Architecture. Table VI compares variants trained from synthetic pretraining onward under a matched 18-epoch budget, together with a full model retrained under the same budget, on the 44 real test scenes of the four-arm main split. These rows are comparable with each other but not with Tables II and III, and we report mean errors, which are sensitive to failure cases. Removing the axis-orientation prompt degrades all four metrics and more than doubles the refined translation error. The other two variants trade translation for rotation: removing the mask-guided arm prompt, or using the feature-pooling baseline with an SE(3) regression head, lowers the refined rotation error but raises the refined translation error by 37% and 23%, respectively. No variant reduces both refined errors.

Consistency losses. Table VII removes one loss at a time (epoch 26, 300 synthetic test scenes of Franka Panda, SO-100, and xArm6) and measures the feed-forward quantity each loss targets, in centimeters after metric scaling: the *lift residual* between the lifted base point and the predicted translation, the RMS *cross-frame spread* of the lifted base point, and the *scene inconsistency*, i.e., the mean pairwise Chamfer distance between non-robot point clouds of different frames. Each loss mainly controls its own target (**bold**): removing $\mathcal{L}_{\text{lift}}$ or $\mathcal{L}_{\text{sc}}$ raises the corresponding error by 64% or 26%, and removing $\mathcal{L}_{\text{xf}}$ more than doubles the cross-frame spread. The losses also interact: $\mathcal{L}_{\text{xf}}$ additionally lowers the lift residual, whereas $\mathcal{L}_{\text{sc}}$ trades some base-point consistency for scene consistency.

Input protocol. We further evaluate the multi-frame protocol on the 44 real test scenes of the four-arm main split. Using a single frame instead of six increases the refined median error from 1.52°/2.11 cm to 2.59°/2.72 cm, showing that multi-frame fusion improves accuracy. To isolate the effect of view diversity, we select three of the six frames with minimal rather than maximal joint-space distance, which halves their pairwise joint-space spread. This increases the feed-forward rotation and translation errors by about 15%, and the error distribution shifts as a whole without introducing new failure cases. View diversity improves accuracy but is not a prerequisite.

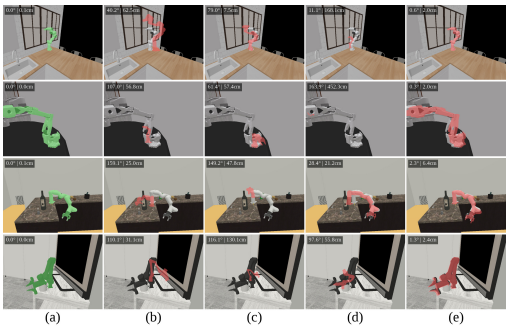


Fig. 4. **Qualitative results on synthetic scenes.** Rows show the four simulated arms; columns show (a) SHEC3R, (b) EasyHeC, (c) EasyHeC++, (d) Kalib, and (e) Hydra, with rotation and translation errors.

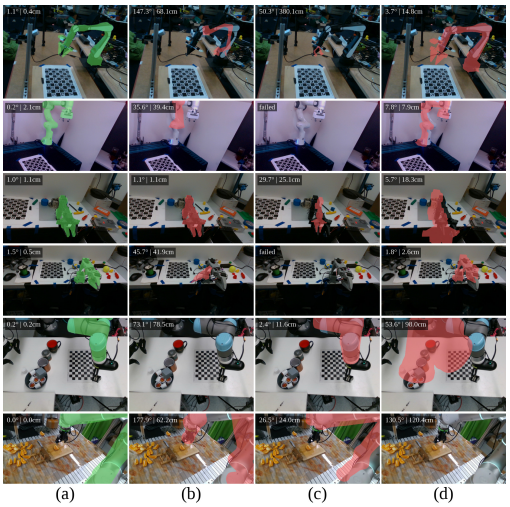


Fig. 5. **Qualitative results on real-world scenes.** Rows show the six real arms; columns show (a) SHEC3R, (b) EasyHeC++, (c) Kalib, and (d) Hydra. "failed" marks unsolved scenes.

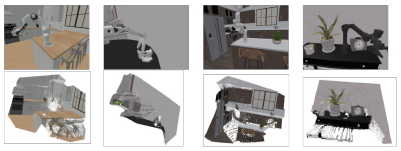
G. Runtime and Robot-Grounded Reconstruction

On an NVIDIA GeForce RTX 2080 Ti GPU, feed-forward prediction takes 0.48 s per scene with $N=6$ frames, and the optional DR refinement takes about 90 s. Grounding DINO + SAM 2 masking adds 0.36 s per frame without human input, so masking and feed-forward inference together take about 2.7 s for a six-frame scene. This excludes data acquisition, i.e., moving the arm through the six joint configurations, which dominates the total calibration time.

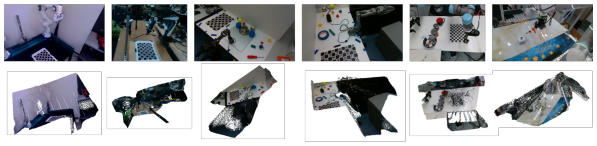
Beyond calibration, the same forward pass predicts dense point maps, which the estimated $\mathbf{T}_{cb}$ transforms into the robot base frame (Fig. 6). They require no additional network inference, and their metric scale is recovered from the same depth frame used for calibration.

VI. LIMITATIONS

SHEC3R has several limitations. Only the feed-forward stage is robot-model-free: refinement uses URDF meshes and joint readings, and metric translation requires one depth frame. Real training data require ChArUco-calibrated extrinsics, so calibration is paid once per robot family, and robots without training data are not supported (Section V-D). Camera intrinsics are inferred implicitly; supplying them would require retraining. Rotation accuracy drops when the



(a) Synthetic: Panda, SO-100, xArm6, WidowX AI



(b) Real: Franka, AIRBOT Play, OpenManipulatorX, SO-101, UR5e, Rizon 4

Fig. 6. **Robot-grounded point clouds.** Input RGB frames (top) and predicted point maps in the robot base frame (bottom), obtained from the same forward pass as the calibration.

base leaves the image (Section V-E), and the near-symmetric xArm6 base can cause 180° yaw flips that refinement cannot correct. Finally, SHEC3R assumes a fixed external camera and robot masks at inference.

VII. Conclusion

We presented SHEC3R, a learned robot-grounded initialization for stationary eye-to-hand calibration. Holding the scenes and the refinement stage fixed shows that the bottleneck is initialization, not refinement. SHEC3R starts 61.5% (sim) and 70.9% (real) of scenes inside the basin of the shared DR stage, against 1–17% for prior initializers, which lifts the share of scenes refined to within 5° from 8–25% to 80.2% and 86.0%. Extending SHEC3R to robots without training data, eye-in-hand cameras, and online recalibration remains future work. We will release RoboEyeCalib, the code, and the model weights.

References

[1] Y. C. Shiu and S. Ahmad, "Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form AX=XB," *IEEE Trans. Robot. Autom.*, vol. 5, no. 1, pp. 16–29, 1989.

[2] R. Y. Tsai and R. K. Lenz, "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration," *IEEE Trans. Robot. Autom.*, vol. 5, no. 3, pp. 345–358, 1989.

[3] Z. Hong, K. Zheng, and L. Chen, "EasyHeC++: Fully automatic hand-eye calibration with pretrained image models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2024.

[4] L. Chen *et al.*, "EasyHeC: Accurate and automatic hand-eye calibration via differentiable rendering and space exploration," *IEEE Robot. Autom. Lett.*, vol. 8, no. 11, pp. 7234–7241, 2023.

[5] D. Allegro, M. Terreran, and S. Ghidoni, "Calib3R: Hand-eye calibration and 3D metric-scaled scene reconstruction with 3D foundation models," arXiv preprint arXiv:2509.08813, 2025.

[6] M. Huber *et al.*, "Hydra: Marker-free RGB-D hand-eye calibration," arXiv preprint arXiv:2504.20584, 2025.

[7] H. Tang *et al.*, "Bi-manual joint camera calibration and scene representation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2026.

[8] S. Xie *et al.*, "Unify robot actions in camera frame," arXiv preprint arXiv:2511.17001, 2025.

[9] T. Tang *et al.*, "Kalib: Easy hand-eye calibration with reference point tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2025, pp. 17 948–17 955.

[10] S. Yang *et al.*, "Robo3R: Enhancing robotic manipulation with accurate feed-forward 3D reconstruction," in *Proc. Robot.: Sci. Syst. (RSS)*, 2026.

[11] Y. Wang *et al.*, "π^3: Permutation-equivariant visual geometry learning," arXiv preprint arXiv:2507.13347, 2025.

[12] J. Lu, F. Richter, and M. C. Yip, "Pose estimation for robot manipulators via keypoint optimization and sim-to-real transfer," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4622–4629, 2022.

[13] T. E. Lee *et al.*, "Camera-to-robot pose estimation from a single image," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020.

[14] A. Khazatsky *et al.*, "DROID: A large-scale in-the-wild robot manipulation dataset," in *Proc. Robot.: Sci. Syst. (RSS)*, 2024.

[15] Y. Labbé *et al.*, "Single-view robot pose and joint angle estimation via render & compare," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021, pp. 1654–1663.

[16] J. Lu *et al.*, "CtRNet-X: Camera-to-robot pose estimation in real-world conditions using a single camera," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2025, pp. 1914–1920.

[17] S. Wang *et al.*, "DUSt3R: Geometric 3D vision made easy," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024.

[18] V. Leroy, Y. Cabon, and J. Revaud, "Grounding image matching in 3D with MASt3R," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2024.

[19] J. Yang *et al.*, "Fast3R: Towards 3D reconstruction of 1000+ images in one forward pass," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2025.

[20] Q. Wang *et al.*, "Continuous 3D perception model with persistent state," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2025.

[21] J. Zhang *et al.*, "MonST3R: A simple approach for estimating geometry in the presence of motion," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2025.

[22] H. Wang and L. Agapito, "3D reconstruction with spatial memory," in *Proc. Int. Conf. 3D Vis. (3DV)*, 2025.

[23] Y. Chen *et al.*, "Human3R: Everyone everywhere all at once," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2026.

[24] J. Wang *et al.*, "VGGT: Visual geometry grounded transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2025.

[25] Open X-Embodiment Collaboration, "Open X-embodiment: Robotic learning datasets and RT-X models," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 6892–6903.

[26] K. Wu *et al.*, "RoboMIND: Benchmark on multi-embodiment intelligence normative data for robot manipulation," in *Proc. Robot.: Sci. Syst. (RSS)*, 2025.

[27] H.-S. Fang *et al.*, "RH20T: A comprehensive robotic dataset for learning diverse skills in one-shot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2024, pp. 653–660.

[28] M. Oquab *et al.*, "DINOv2: Learning robust visual features without supervision," *Trans. Mach. Learn. Res.*, 2024.

[29] Y. Zhou *et al.*, "On the continuity of rotation representations in neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 5738–5746.

[30] S. Laine *et al.*, "Modular primitives for high-performance differentiable rendering," *ACM Trans. Graph.*, vol. 39, no. 6, 2020.

[31] S. Tao *et al.*, "ManiSkill3: GPU parallelized robotics simulation and rendering for generalizable embodied AI," in *Proc. Robot.: Sci. Syst. (RSS)*, 2025.

[32] S. Nasiriany *et al.*, "RoboCasa: Large-scale simulation of household tasks for generalist robots," in *Proc. Robot.: Sci. Syst. (RSS)*, 2024.

[33] A. Szot *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021.

[34] J. Straub *et al.*, "The Replica dataset: A digital replica of indoor spaces," arXiv preprint arXiv:1906.05797, 2019.

[35] E. Kolve *et al.*, "AI2-THOR: An interactive 3D environment for visual AI," arXiv preprint arXiv:1712.05474, 2017.

[36] M. Deitke *et al.*, "ProcTHOR: Large-scale embodied AI using procedural generation," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2022.

[37] M. Deitke *et al.*, "RoboTHOR: An open simulation-to-real embodied AI platform," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[38] S. Liu *et al.*, "Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2024.

[39] N. Ravi *et al.*, "SAM 2: Segment anything in images and videos," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2025.