EasyHeC++: Fully Automatic Hand-Eye Calibration with Pretrained Image Models

Linghao Chen[1,2*] Kangfu Zheng[3*] Zhengdong Hong[1,2*] Xiaowei Zhou[1] Hao Su[2]
[1] Zhejiang University [2] UC San Diego [3] Tsinghua University

***Abstract*— Hand-eye calibration plays a fundamental role in robotics by directly influencing the efficiency of critical operations such as manipulation and grasping. In this work, we present a novel framework, EasyHeC++, designed for fully automatic hand-eye calibration. In contrast to previous methods that necessitate manual calibration, specialized markers, or the training of arm-specific neural networks, our approach is the first system that enables accurate calibration of any robot arm in a marker-free, training-free, and fully automatic manner. Our approach employs a two-step process. First, we initialize the camera pose using a sampling or feature-matching-based method with the aid of pretrained image models. Subsequently, we perform pose optimization through differentiable rendering. Extensive experiments demonstrate the system's superior accuracy in both synthetic and real-world datasets across various robot arms and camera settings. The code will be publicly available upon the publication of this paper.**

I. Introduction

Hand-eye calibration is a fundamental problem in robotics. It connects the vision system and the robot arm system by transforming the perception of the camera into the robot's coordinate system. This is crucial for many robotic applications, such as robotic grasping [1], [2], robotic manipulation [3], [4], and robotic assembly [5].

Traditionally, the hand-eye calibration problem is addressed by using a marker [6], [7], [8] to assist computing the camera pose by solving a $AX = XB$ or $AX = YB$ equation [7], [8], [9], [10], [11]. These methods not only necessitate the placement of a high-quality marker in the scene but also require the manual selection of a series of joint poses. This manual process is time-consuming, and not user-friendly, thereby restricting their applicability in real-world lab and household scenarios.

Recently, learning-based methods have been proposed to address the hand-eye calibration problem. These methods typically involve employing a neural network to either directly regress the camera pose [12], [13] or detect keypoints of the robot arm [14], [15], [16], followed by solving the camera pose using the Perspective-n-Point (PnP) algorithm [17]. The performance of these methods is limited by the quality and quantity of the training data. Moreover, one trained model can only be applied to a single type of robot arm. If the need arises to calibrate a different type of robot arm, it requires starting the process again, which involves collecting new data and training a new model.

Recent works [18], [19] propose to use differentiable rendering to optimize the camera pose with a pixel-wise mask loss and deliver superior accuracy. These methods get rid of accurate marker placement and intricately selected joint poses, thus highly suitable in household scenarios, especially when re-calibration is needed at deployment time. However, they still require a rough pose as initialization and a mask as supervision. The mask segmentation network requires training for each type of robot arm. In addition, the pose initialization is either manually set or obtained by another neural network. Both factors have increased human efforts and training costs, prohibiting these methods from being fully automatic and widely adopted.

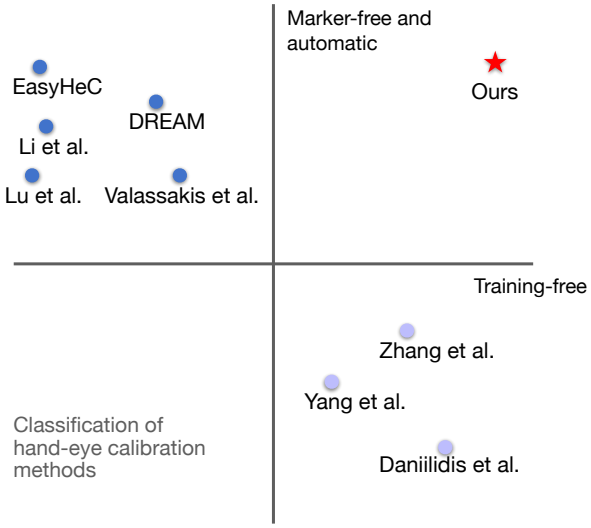


Fig. 1: **Comparison between our method and previous methods.** Our method not only delivers high accuracy but also is fully automatic, marker-free, and training-free.

In this paper, we introduce EasyHeC++ offering the following features:

- Accurate (3mm error) and fully automatic hand-eye calibration system.
- The first method without any training cost or manual annotation for any type of robot arm.
- Fast setup, 15 minutes for a new robot arm, and 5 minutes for re-calibration.
- Both eye-to-hand and eye-in-hand configurations are supported.

Our core idea is to integrate the outstanding generalization abilities of pre-trained image models with the precision derived from the differentiable-rendering-based pose optimization. This innovative design simplifies the prerequisites

*Equal contribution. Corresponding author: Hao Su.

by requiring only a real robot arm, a camera, and the robot arm model. Consequently, this approach significantly diminishes human intervention and training costs, rendering our method more practical for real-world applications.

Our framework consists of two main components: 1) pose initialization and 2) pose optimization. In the pose initialization phase, we utilize either a sampling-based method or a feature-matching-based method to initialize the camera pose for the initial calibration and subsequent re-calibrations, respectively. The sampling-based initialization selects a pose from a set of densely sampled camera poses on a hemisphere. We then refine this pose using differentiable rendering under the supervision of the masks generated by a text-prompted segmentation model [20]. On the other hand, the feature-matching-based initialization leverages the historical calibrated image-pose pairs to infer an initial camera pose. These methods, in contrast to training a neural network or manually setting the camera pose, provide a more automatic approach and are not constrained by the type of robot arm. Next, in the pose optimization phase, we follow the methods outlined in [19], which optimizes the camera pose with a pixel-wise mask loss and uses a space exploration module to search for the most informative joint pose, yielding more precise calibration results. In this phase, the mask supervision is generated by a pre-trained open-world segmentation model [21], using the projected robot arm model as prompts. Leveraging the kinematics model of the robot arm to guide the pretrained image models fully capitalizes on their generalization ability and allows us to generate an accurate mask without the need for any manual prompt annotation or training tailored to a specific type of robot arm. After the camera pose is solved, we incorporate the calibrated image-pose pair into the database. This information will serve as the reference for pose initialization in subsequent re-calibrations using the feature-matching-based method.

We evaluate the proposed method across various types of robot arms in synthetic and real-world scenes. The results demonstrate that our method outperforms all the previous methods in terms of accuracy and automation under both eye-to-hand and eye-in-hand settings. To the best of our knowledge, this is the first work that achieves fully automatic hand-eye calibration, eliminating the necessity for training, manual annotation, or markers. In a commitment to contribute to the robotics community, we are dedicated to open-sourcing our system for wider accessibility and benefit.

II. Related Work

A. Hand-Eye Calibration

Hand-eye calibration is the problem of obtaining the transformation matrix between the camera and the robot reference frames, which involves two camera settings: eye-to-hand calibration and eye-in-hand calibration. In eye-to-hand calibration, the camera is stationary relative to the robot base. Conversely, for eye-in-hand calibration, the camera is stationary relative to the robot's end-effector.

Traditional methods. Traditional methods employ a marker to solve AX=XB hand-eye formulation for eye-to-hand calibration [7], [8], [9], [10] or AX=YB robot-world-hand-eye formulation for eye-in-hand calibration [22]. Several recent approaches have integrated these formulations with active next-best-view selection to make the calibration process more automated and precise. Zhang et al. [23] proposed an online estimated discrete viewing quality field to represent the calibration quality of selected camera views. Yang et al. [24] introduced uncertainty reduction to guide the robot pose selection. However, these approaches are highly affected by the visibility and quality of the markers, which limits their applicability and increases the difficulty for users to utilize them.

Learning-based methods. Recent works proposed learning-based or marker-free methods to solve hand-eye calibration. For eye-to-hand setting, DREAM [15] introduced a two-step framework, which first utilizes a deep neural network to detect 2D projections of keypoints from the RGB image of the robot, and then recovers the camera poses using Perspective-n-Point (PnP) algorithm [17]. Similarly, Lu et al. [16] proposed to first find the optimal set of keypoints on the robot through an iterative approach using DNNs, and subsequently to estimate pose utilizing the PnP algorithm.

Optimization-based methods. Recently, Lu et al. [18] proposed to use differentiable rendering to optimize the camera pose with a pixel-wise mask loss as the objective function. EasyHeC [19] further designed an uncertainty-based space exploration module to search for the next best informative joint pose for more accurate calibration results for the eye-to-hand setting. For the eye-in-hand setting, Valassakis et al. [12] trained a simple neural network and directly regressed the camera pose from images captured by a mounted camera. For both eye-to-hand and eye-in-hand settings, Li et al. [13] proposed to detect the robot base and align it with a point cloud to compute the camera pose. All the existing methods either necessitate manual calibration with specialized markers or require the training of arm-specific neural networks. As a comparison, we are the first hand-eye-calibration system in a marker-free, training-free, and fully automatic manner.

Difference from EasyHeC Our method is built upon EasyHeC, however, there are several key differences. **(1)** EasyHeC requires mask and pose networks re-training for each new type of robot arm, while our method is training-free and can be applied to any robot arm. **(2)** EasyHeC is designed only for one-time calibration, while we support fast recalibration after a camera repositioning. **(3)** EasyHeC only supports the eye-to-hand setting, while we also support the eye-in-hand setting.

B. Visual Localization

Visual localization is an important computer vision task that aims to estimate the camera pose of a new image given a known scene representation. The scene is usually represented as the reconstruction results of Structure-from-Motion [25] using feature-matching-based methods [26], [27]. Then the

core problem of visual localization becomes finding the correspondences between the pixels in the 2D image and the 3D points in the scene. HLoc [28] proposed to address the visual localization problem by image retrieval and lifting 2D-2D matchings to 2D-3D matchings in a coarse-to-fine manner. OnePose [29] proposed to first reconstruct a semi-dense point cloud representation for the 3D object and then train a neural network to directly match the 2D pixel to the 3D point cloud. OnePose++ [30] proposed to substitute the COLMAP [25] with a learning-based feature matching approach [31] to improve the performance on textureless objects. However, these methods are not suitable for our scenarios. The reconstruction in their methods has no canonical space, whereas the robot arm has a pre-defined canonical space. Moreover, the robot arm has a known object shape, making aligning them highly reliant on the reconstructed shape quality and camera pose accuracy. The most relevant work in this field is MeshLoc [32], which proposed to use a mesh-based representation to avoid feature matching between database images. In terms of hand-eye calibration, the robot mesh is usually off-the-shelf and can be easily obtained.

C. Image Models for Segmentation

Traditional learning-based methods using networks like [33], [34] are largely confined by their generalizability in tasks and data distributions beyond those seen during training. In our task, as we aim to segment the highly-articulated robotic arm, those models require us to collect extra robot arm images for training, which is laborious. The current trend in large vision language models like Segment Anything (SAM) [21] has enabled precise zero-shot image segmentation. Boosted by its scaled model size and abundant text corpora from the web, SAM [21] and its follow-up works [21], [35], [36], [37] dominate image segmentation by its superiority in quality, speed, and generalizability. Another virtue of large vision language models is they are designed and trained to be promotable. Grounded-Segment-Anything [20] further broadens the scope of the application by supporting image, text, and speech inputs. In our pipeline, we use the kinematics model of the robotic arm to guide the SAM model [20] to generate masks without the need for manually labeled prompts.

III. Methods

A. Background

In this work, we aim to address the hand-eye calibration problem under two settings: eye-to-hand calibration and eye-in-hand calibration. We represent the relative pose between the camera and the robot base as T_{cb} and the relative pose between the camera and the end-effector as T_{ce}.

Our method is built on EasyHeC, which addresses the eye-to-hand calibration problem iteratively. Each iteration includes two main components: differentiable-rendering-based camera pose optimization and consistency-based joint space exploration. The differentiable-rendering-based optimization uses a pixel-wise rendering mask loss to optimize the camera pose T_{cb} as follows:

$$L(\xi_{cb}) = \left(\min \left(1, \sum_{l} \pi \left(\exp\left(\xi_{cb}\right) T_{bl} l \right) \right) - M \right)^2, \quad (1)$$

where $\xi_{cb} \in \mathfrak{se}(3)$ is the exponential coordinate of the relative pose between the camera and the robot base, π is a differentiable mask renderer, T_{bl} is the relative pose between the base link and the link l, computed from forward kinematics, and M is the observed mask, inferred from the RGB image captured by the camera c.

After the optimization of each iteration, the consistency-based joint space exploration is performed. This process samples a bunch of joint poses in the simulator and identifies the most informative one to improve the accuracy of the calibration results. Then, the robot arm moves to the next joint pose and the optimization process is performed again on all the collected images. This process is repeated until the number of iterations reaches a pre-defined maximum number. We call this number of iterations as the space exploration iterations in the following sections. To learn more details, please refer to [19].

B. Overview

As shown in Fig. 2, EasyHeC++ aims to solve the hand-eye calibration problem in a fully automatic manner. Notably, we address not only the first-time calibration but also the subsequent re-calibration after the camera movement. At each time of calibration, we first initialize the camera pose either via a sampling-based method (Sec. III-C.1) or a feature-matching-based method using the existing image-pose pairs in the database (Sec. III-C.2). Next, we perform the pose optimization in the same way as EasyHeC, using differentiable-rendering-based optimization and consistency-based joint space exploration. The main difference is that we design an AutoSAM module to automatically predict segmentation masks (Sec. III-D) as the supervision. After each time of calibration, the image-pose pair is added to the database as the reference images for the initialization in the subsequent re-calibrations. We use eye-to-hand calibration as the default setting in Sec. III-C and Sec. III-D and finally discuss the eye-in-hand calibration in Sec. III-E.

C. Automatic Pose Initialization

The hand-eye-calibration accuracy is highly dependent on the quality of the pose initialization. In the previous work [19], the pose initialization is obtained by a neural network. The network has to be trained on synthetic data and fine-tuned on real data for each type of robot arm, which may suffer from the sim-to-real domain gap and not be user-friendly due to the additional effort of per-arm data annotation and training. In this work, we aim to initialize the pose in an automatic and training-free manner.

1) Sampling-based Pose Initialization: Given a robot arm that has never been calibrated before, we aim to estimate an initial camera pose T_{cb}^{init} for the following differentiable-rendering-based pose optimization process.

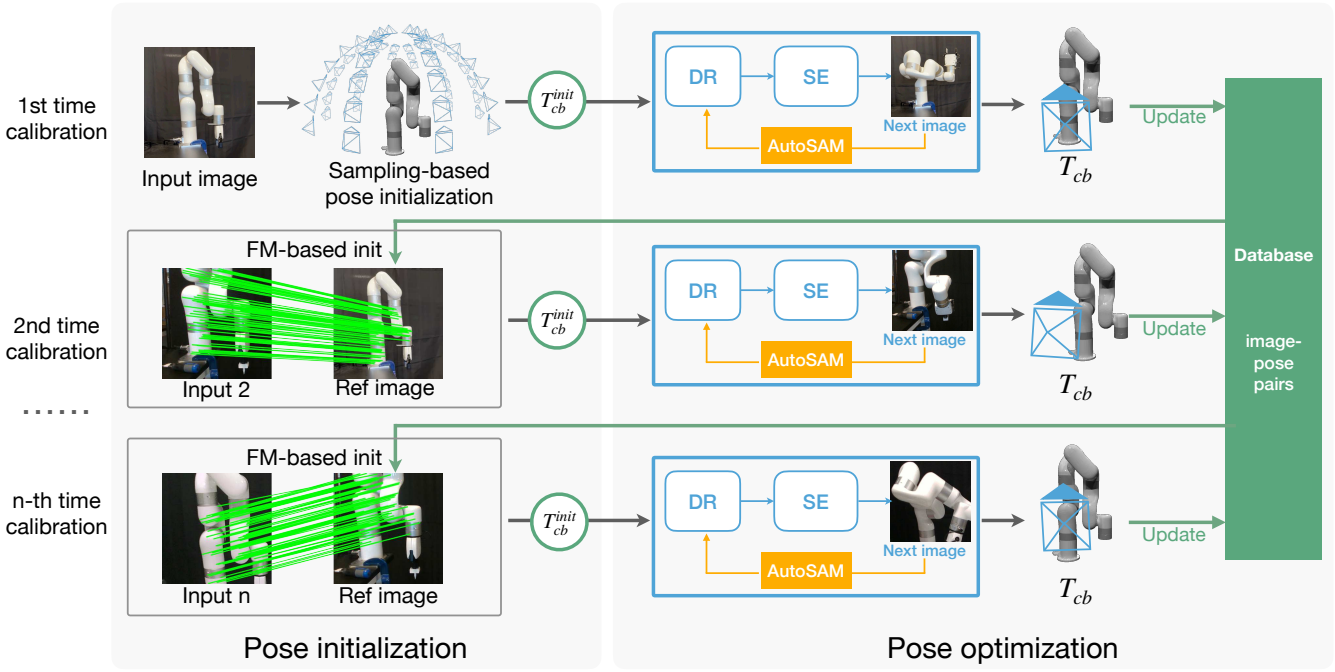


Fig. 2: **EasyHeC++ architecture.** We consider not only single-instance calibration but also recalibration after camera movement. At each time of calibration, EasyHeC++ consists of two main components: pose initialization and pose optimization. At the first time of calibration, we use a sampling-based pose initialization module to initialize a rough camera pose T_{cb}^{init}, while in subsequent re-calibrations, we use a feature matching (FM)-based module to initialize the camera pose, using the historical image-pose pairs in the database. Then we run pose optimization by first using a differentiable renderer (DR) to optimize the camera pose and then running a space exploration (SE) to obtain the next joint pose to increase the accuracy. In this process, AutoSAM is proposed to predict mask as the supervision to the DR process. After solving the camera pose T_{cb}, we add the image-pose pair to the database.

The basic idea is to generate segmentation masks using the large segmentation model, GroundedSAM [20], with "robot arm" as the text prompt and enumerate the camera poses to find the one with the highest similarity to the observed mask. Concretely, we sample a bunch of camera poses on a hemisphere as in Fig. 2. Then we render the robot arm model to obtain the rendered mask and compute the mask IoU between the rendered and observed masks for each camera pose. Then we select the camera pose with the highest mask IoU and further refine it using differentiable rendering in Eq. 1 to obtain the final initial pose T_{cb}^{init}, where the mask predicted by GroundedSAM is used to compute the mask loss.

In practice, the process of sampling different joint poses is only required for the first time. In the subsequent re-calibrations, we adopt the feature-matching-based initialization introduced in the next subsection, which only requires a single joint pose as input.

2) Feature-Matching-based Pose Initialization: As shown in Fig. 2, after each time of calibration, we update the database with the image at the initial joint pose and the solved camera pose. When re-calibration is needed due to the camera movement, we utilize the existing image-pose pairs to initialize the camera pose via a feature-matching technique as in [32].

Specifically, for an image-pose pair $\{I^r, T_{cb}^r\}$ in the database, we can obtain $\{p^r, P^c\}$ as the correspondences between the 2D pixels and the 3D point cloud in the robot arm canonical space. Then, given the target image I^t, we can adopt a pre-trained dense feature-matching network to predict the pixel correspondences $\{p^r, p^t\}$ between the reference image I^r and the target image I^t. Then the 2D-2D correspondences are lifted to 2D-3D correspondences $\{p^t, P^c\}$ to solve the camera pose T_{cb}^t.

Benefiting from the nature of the feature-matching network which operates on local patches, this method is not restricted by the type of robot arm and is thus fully automatic.

D. Automatic Segmentation Prediction

Predicting an accurate segmentation mask of the robot arm plays a crucial role in the differentiable-rendering-based optimization process since its quality directly impacts the resulting calibration accuracy. Previous methods [18], [19] train neural networks [38], [34] to predict this segmentation mask, which is not only time-consuming but also restricted to a specific type of robot arm. The recent large vision language models [36], [37], [21], [20] have shown impressive performance in the open-world image segmentation task, but they either require a manual prompt as input [36], [21] or deliver inferior mask predictions [20].

In this work, we design a module called AutoSAM, which uses the kinematics model of the robot arm as a guide to SAM [21] to generate the masks in a fully automatic and

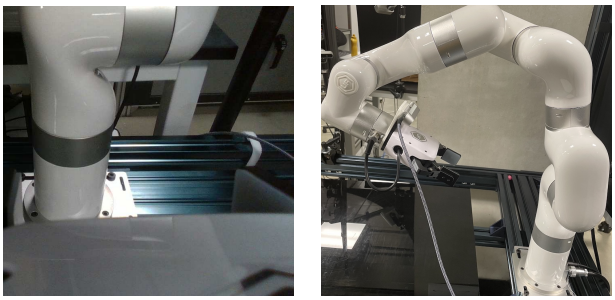
training-free manner. The basic idea is to use the initialized camera pose or the optimized camera pose in the last space exploration iteration to project the robot arm model to the current frame. Then the projection is used as a prompt to generate the segmentation mask as $M_l = \text{SAM}(I_t, \Pi(q_t, l; T_{cb}^{t-1}))$, where M_l is the predicted mask of link l, I_t is the RGB image at the current joint pose q_t, Π is a projection operation producing a 2D bounding box for link l, and T_{cb}^{t-1} is the camera pose solved in the last space exploration iteration or the initialized camera pose in the first iteration. Then, we combine all the masks of the links to obtain the final mask M. Practically, in addition to link-wise masks, we also use the 2D bounding box of every 2 adjacent links as the prompt to further improve the mask quality. As shown in Sec. IV-D, this link-wise prompted SAM can generate superior mask quality than the previous methods while requiring zero training cost.

E. Eye-In-Hand Camera Pose Optimization

Eye-in-hand calibration is another setting under the hand-eye calibration problem, in addition to the to-hand setting. Different from previous eye-in-hand works that rely on a marker [23], [24] or use a neural network to predict the camera pose [12], we follow the previous work, EasyHeC [19], which uses a pixel-wise mask loss to optimize the camera pose. Specifically, the loss function is defined as follows:

$$L(T_{ce}) = \left(\min \left(1, \sum_l \pi \left(\exp \left(\xi_{ce} \right) T_{eb} T_{bl} l \right) \right) - M \right)^2, \quad (2)$$

where T_{eb} and T_{bl} are the relative pose between the end-effector and the robot base and the relative pose between the robot base and the link l, respectively, computed from forward kinematics, ξ_{ce} is exponential coordinate of the relative pose between the camera and the end-effector, and M is the observed mask generated by AutoSAM.



(a) (b)

Fig. 3: **Example images for our method under the eye-in-hand setting.** (a) is the image captured by the in-hand camera and (b) is the image captured from the spectator's view for illustration.

Because of the nature of the eye-in-hand setting, it is not guaranteed that the robot arm is visible in the image at a random joint pose. Thus we propose to use a different joint pose sampling strategy from the eye-to-hand setting to ensure the robot arm is visible in the image. Specifically, instead of sampling the angle of each joint independently as in [19], we first sample the end-effector pose to ensure the end-effector is oriented towards the robot base link properly, and then we compute the joint poses using inverse kinematics. An example image is shown in Fig. 3.

Moreover, we observe that the gripper often occupies a large region in the image. This region is usually not informative for camera pose optimization. Besides, this part is an edge area with low undistortion quality in the image, so including this part in optimization could lead to a significant accuracy drop. To address this problem, we propose to ignore this region in the optimization process with the following mask loss:

$$L(T_{ce}) = \sum_{p \notin O} \left(\min \left(1, \sum_l \pi \left(\exp \left(\xi_{ce} \right) T_{eb} T_{bl} l \right) \right) - M \right)_p^2, \quad (3)$$

where p and O are a pixel and the region of the gripper in the image, respectively.

F. Implementation Details

In the sampling-based pose initialization module, the distance between the camera and the robot arm is fixed to 1 meter, and the elevation angle is sampled from 0 degrees to 70 degrees with an interval of 10 degrees. The azimuth angle is sampled from 0 degrees to 360 degrees with an interval of 30 degrees. In the feature-matching-based pose initialization module, we first use image retrieval [39] to find the image that is the most similar to the target image and use the feature-matching-based method to initialize the pose. we use DKM [40] as our feature-matching network since it handles texture-less and specular robot arm surfaces quite well. Then we use PnP with RANSAC to solve the pose of the target image based on the correspondences. Other hyperparameters in the differentiable-rendering-based pose optimization and consistency-based joint space exploration are the same as EasyHeC [19]. For the initial calibration in the eye-in-hand setting, instead of using the sampling-based pose initialization module, we manually initialize the camera pose since this pose usually demonstrates much smaller variation compared to the eye-to-hand setting. Other operations remain the same as in the eye-to-hand setting.

IV. Experiments

A. Evaluation on Synthetic Datasets

1) Eye-to-Hand Setting: In this section, we evaluate our method using a synthetic dataset proposed in [19]. The dataset consists of 100 scenes captured under different camera poses under the eye-to-hand setting. We conduct a comparative analysis, evaluating the performance of our method in comparison to previous approaches. Following the evaluation protocols in [19], we only evaluate the scenes where [8] and [15] successfully solve the camera pose.

Tab. I and Tab. II present the evaluation results of rotation and translation errors, respectively, where our method consistently outperforms previous approaches. Even using a single view, our method surpasses the performance of the prior work [19], attributed to the precision of our pose initialization Despite both EasyHeC [19] and our method using the same

Method	#views	Rotation error (°)				
Marker-based [8]	20	0.870				
DREAM [15]	1 ∼ 5	1.924	1.240	0.981	0.764	0.704
EasyHeC [19]	1 ∼ 5	0.322	0.128	0.109	0.097	0.081
Ours	1 ∼ 5	**0.246**	**0.076**	**0.064**	**0.058**	**0.045**

TABLE I: **Rotation error evaluation results on the xArm synthetic dataset.**

Method	#views	Translation error (cm)				
Marker-based	20	2.000				
DREAM	1 ∼ 5	0.529	0.473	0.374	0.347	0.303
EasyHeC	1 ∼ 5	0.488	0.298	0.252	0.206	0.206
Ours	1 ∼ 5	**0.318**	**0.176**	**0.159**	**0.137**	**0.135**

TABLE II: **Translation error evaluation on the xArm synthetic dataset.**

differentiable-rendering-based optimization for camera pose resolution, our approach achieves superior accuracy. This superiority is attributed to our method's utilization of the robot arm's kinematics model and a large pre-trained segmentation model to predict segmentation masks, which proves more accurate than the masks generated in [19]. Specifically, our method achieves a translation error of 0.135cm with 5 joint poses, outperforming EasyHeC [19] which records a 0.206cm error under similar conditions.

Method	#views	Rotation error (°)				
Zhang et al. [23]	10	0.148				
Valassakis et al. [12]	1	4.4				
Ours	1 ∼ 5	**0.685**	**0.204**	**0.145**	**0.130**	**0.117**

TABLE III: **Rotation error evaluation results on the xArm synthetic dataset.**

Method	#views	Translation error (cm)				
Zhang et al. [23]	10	0.315				
Valassakis et al. [12]	1	1.340				
Ours	1 ∼ 5	**0.503**	**0.179**	**0.128**	**0.117**	**0.112**

TABLE IV: **Translation error evaluation results on the xArm synthetic dataset.**

2) Eye-in-Hand Setting: Similar to the eye-to-hand setting, we evaluate our method on a synthetic dataset under the eye-in-hand setting and compare it to previous methods. The evaluation involves 50 different camera poses, and we simulate the calibration process using SAPIEN [41]. For the approach proposed in Valassakis et al. [12], we synthesize 10000 images using a xArm7 robot arm and train their model. For Zhang et al. [23], we reproduce their method with a chessboard of 4×5 grids with a 5cm grid size and images captured at 10 different joint poses. Compared to [12], which trains a neural network to regress the camera pose, our method not only requires no training but also achieves better accuracy. Furthermore, in comparison to [23], our method achieves better accuracy with fewer joint poses.

B. Evaluation on Real Dataset

We conducted a comparative evaluation of our method against several previous approaches using the real-world Baxter dataset [18]. This dataset comprises 100 images captured under the same camera pose but with 20 different joint poses. The evaluation metrics include the percentage of correct keypoints (PCK) for both 2D and 3D, as presented in Tab. V and Tab. VI, respectively. The tables demonstrate that our method outperforms previous approaches in both 2D and 3D PCK, particularly under small thresholds. For instance, when using 3 views, our method achieves a 2D PCK of 0.5 and 0.7 with 10 px and 20 px thresholds, respectively, compared to only 0.05 and 0.55 for EasyHeC [19]. Despite EasyHeC training a segmentation model on synthetic data with data augmentation, its generalization to real-world images remains challenging. In contrast, our method leverages the generalization capability of the pre-trained image model to predict the segmentation mask. This not only eliminates the need for time-consuming and intricate data augmentation during training but also yields superior mask quality, resulting in higher PCK values.

Method	PCK 2D						
	10px	20px	30px	40px	50px	100px	150px
DREAM [15]	-	0.16	0.23	0.29	0.33	0.52	0.62
OK [16]	-	0.34	0.54	0.66	0.69	0.88	0.93
IPE [18] (box)	-	-	-	-	0.65	0.94	0.95
IPE [18] (cylinder)	-	-	-	-	0.80	0.91	0.93
IPE [18] (CAD)	-	-	-	-	0.74	0.90	0.94
EasyHeC (1view)	0.1	0.35	0.55	**0.75**	0.90	**0.95**	0.95
EasyHeC (2views)	0.15	0.40	0.75	**0.95**	**1.00**	**1.00**	**1.00**
EasyHeC (3views)	0.05	0.55	**0.85**	**1.00**	**1.00**	**1.00**	**1.00**
Ours (1view)	**0.25**	**0.5**	**0.75**	**0.75**	0.85	**0.95**	**1.00**
Ours (2views)	**0.5**	**0.55**	**0.9**	0.9	0.9	**1.00**	**1.00**
Ours (3views)	**0.5**	**0.7**	**0.85**	0.95	**1.00**	**1.00**	**1.00**

TABLE V: **2D PCK evaluation results on the Baxter dataset.** 2D PCK scores are given at different thresholds.

Method	PCK 3D					
	2cm	5cm	10cm	20cm	30cm	40cm
DREAM [15]	0.01	0.08	0.32	0.43	0.54	0.66
OK [16]	0.10	0.34	0.54	0.66	0.69	0.88
IPE [18] (box)	-	-	0.8	0.95	0.95	0.95
IPE [18] (cylinder)	-	-	0.71	0.93	0.94	0.95
IPE [18] (CAD)	-	-	0.78	0.93	0.97	**1.00**
EasyHeC (1view)	0.10	0.65	0.90	**1.00**	**1.00**	**1.00**
EasyHeC (2views)	0.15	**0.80**	**0.95**	**1.00**	**1.00**	**1.00**
EasyHeC (3views)	0.15	**0.80**	0.90	**1.00**	**1.00**	**1.00**
Ours (1view)	**0.15**	**0.75**	0.95	**1.00**	**1.00**	**1.00**
Ours (2views)	**0.2**	0.6	**0.95**	**1.00**	**1.00**	**1.00**
Ours (3views)	**0.3**	0.65	0.90	**1.00**	**1.00**	**1.00**

TABLE VI: **3D PCK evaluation results on the Baxter dataset.**

Method	DREAM	EasyHeC	EasyHeC (SAM)	Ours
Error (cm)	1.5	0.4	**0.3**	**0.3**

TABLE VII: **Real-world error high-precision targeting experiment under the eye-to-hand setting.**

C. Real-world Evaluations

In addition to the real-world Baxter dataset [18], we evaluated our method under a real-world setup. Our experimentation involved using a xArm7 robot arm with a RealSense

Method	Zhang et al. [23]	Valassakis et al. [12]	Ours
Error (cm)	1.35	4.30	**0.31**

TABLE VIII: **Real-world error high-precision targeting experiment under the eye-in-hand setting.** Errors are computed across 10 tipping trails.

camera, positioned either on a nearby tripod or mounted on the end-effector for the eye-to-hand and eye-in-hand settings, respectively. After applying our calibration method, we follow the procedure outlined in previous work [19] for further evaluation. This involves transforming the corner of an ArUco marker to the robot base coordinate system, tipping it, and manually measuring the error between the tip and corner. The results are presented in Tab. VII and Tab. VIII, where our method achieves the lowest error in both settings. In the eye-to-hand setting, it is noteworthy that both EasyHeC (SAM) and our method utilize the SAM model for mask prediction. However, they require manually annotated bounding-box prompts, while our method does not, enhancing its automation.

D. Ablation Study

Automatic pose initialization. We conducted a comparison of different pose initialization methods between EasyHeC [19] and our proposed approach. The results are shown in Tab. IX. EasyHeC trained a PVNet [42] on synthetic data to initialize the camera pose at the robot arm's zero joint pose. Both our sampling-based and feature-matching-based pose initialization achieve superior accuracy compared to PVNet. The sampling-based initialization proves robust to mask quality, while the feature-matching-based method efficiently utilizes existing calibration results to initialize the pose. Notably, the sampling-based method demonstrates higher accuracy but is more time-consuming, requiring the robot arm to be driven to multiple joint poses, taking approximately 10 minutes. In contrast, the feature-matching-based method only requires the robot arm to have the same joint pose as the images in the database, avoiding the need for arm movement and proving more efficient.

Method	Rot. error (°) ↓	Trans. error (cm) ↓	Time ↓
EasyHeC (PVNet)	3.94	2.40	10h+/100ms
Ours (Sampling)	0.10	0.26	0/∼10min
Ours (Feat. matching)	2.33	1.80	0/6s

TABLE IX: **Ablation study for pose initialization.** Errors on the xArm eye-to-hand synthetic dataset. Training time/inference time are reported on an RTX 4090 GPU.

Different prompts to the SAM model. In this ablation study, we compare different prompts to the SAM model on the xArm synthetic dataset. The qualitative and quantitative results are shown in Fig. 4 and Tab. X, respectively. While the most straightforward prompt is to use a single bounding box for the entire robot arm, we observe that this prompt lacks accuracy, as shown in Fig. 4(a). This is because the space exploration module tends to produce a contorted joint pose, making it challenging for the SAM model to generate precise masks. Additionally, real-world scenarios may involve occlusions from the table or other objects, as well as unwanted attachments on the robot arm (e.g., an in-hand camera), further diminishing the accuracy of the single bounding box prompt. Combining the single bounding box prompt with a center point prompt can even lead to a reduction in accuracy, as depicted in Fig. 4(b). Although using per-link bounding boxes, as shown in Fig. 4(c) and Fig. 4(d), can enhance accuracy, it remains somewhat unstable, occasionally missing connectors between adjacent links. The most accurate prompt involves using bounding boxes for each link and connectors between each pair of adjacent links, as demonstrated in Fig. 4(e) and Fig. 4(f). This comprehensive prompt configuration yields the highest accuracy in generating precise masks.

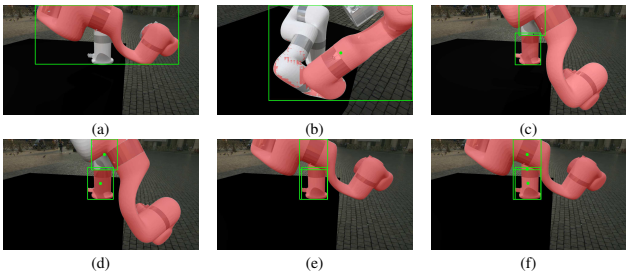


Fig. 4: **Ablation study on different types of prompts to the SAM model.** (a) A single bounding box of the whole robot arm as the prompt. (b) A single bounding box of the robot arm and its center point as the prompt. (c) Per-link bounding boxes as the prompt. (d) Per-link bounding boxes and their center points as the prompt. (e) Bounding boxes of each link and each connector as the prompt. (f) Bounding boxes of each link and each connector and their center points as the prompt. For clarity, we do not show all the prompts from (c) to (f).

Method	Box prompt	Point prompt	Connector	Mask IoU ↑
EasyHeC	-	-	-	96.3
Ours (a)	Single	w/o	w/o	92.3
Ours (b)	Single	w/	w/o	93.0
Ours (c)	per-link	w/o	w/o	93.7
Ours (d)	per-link	w/	w/o	94.2
Ours (e)	per-link	w/o	w/	**98.2**
Ours (f)	per-link	w/	w/	**98.2**

TABLE X: **Mask IoU comparison with different types of prompts to the SAM model tested on the xArm eye-to-hand synthetic dataset.** EasyHeC costs over 20 hours for training on an RTX 4090 GPU, while Ours requires no training. The indication of Roman numerals (a)-(f) are shown in Fig. 4.

V. Conclusion

In this work, we proposed EasyHeC++, which can calibrate any robot arm in a marker-free, training-free, and fully automatic manner. Our main approach consists of a pose initialization phase and a pose optimization phase. By

integrating the generalization ability of pretrained image models and the accuracy of optimization-based methods, EasyHeC++ achieves a fully automatic pipeline. Experiments show that our method produces superior accuracy and degree of automation in both synthetic and real-world datasets for different robot arms and camera settings. This work opens up more possibilities for lab and household applications [1], [2], [3], [4], [5], [43], [44] that require hand-eye calibration to reduce the sim-to-real gap, such as robot manipulation and grasping.

REFERENCES

[1] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[2] L. Chen, Y. Song, H. Bao, and X. Zhou, "Perceiving unseen 3d objects by poking the objects," in *ICRA*, 2023, pp. 4834–4841.

[3] Z. Jia, F. Liu, V. Thumuluri, L. Chen, Z. Huang, and H. Su, "Chain-of-thought predictive control with behavior cloning," in *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*.

[4] B. An, Y. Geng, K. Chen, X. Li, Q. Dou, and H. Dong, "Rgbmanip: Monocular image-based robotic manipulation through active object pose estimation," *arXiv preprint arXiv:2310.03478*, 2023.

[5] S. Zobov, F. Chervinskii, A. Rybnikov, D. Petrov, and K. Vendidandi, "Auto-assembly: a framework for automated robotic assembly directly from cad," *arXiv preprint arXiv:2301.02643*, 2023.

[6] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[7] J. Ilonen and V. Kyrki, "Robust robot-camera calibration," in *ICAR*. IEEE, 2011, pp. 67–74.

[8] R. Y. Tsai, R. K. Lenz *et al.*, "A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration," *IEEE Transactions on robotics and automation*, vol. 5, no. 3, pp. 345–358, 1989.

[9] F. C. Park and B. J. Martin, "Robot sensor calibration: solving ax= xb on the euclidean group," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 5, pp. 717–721, 1994.

[10] K. Daniilidis, "Hand-eye calibration using dual quaternions," *The International Journal of Robotics Research*, vol. 18, no. 3, pp. 286–298, 1999.

[11] R. Horaud and F. Dornaika, "Hand-eye calibration," *The International Journal of Robotics Research*, vol. 14, no. 3, pp. 195–210, 1995.

[12] E. Valassakis, K. Dreczkowski, and E. Johns, "Learning eye-in-hand camera calibration from a single image," in *CoRL*. PMLR, 2022, pp. 1336–1346.

[13] L. Li, Y. Du, X. Yang, R. Wang, and X. Zhang, "Look at robot base once: Hand-eye calibration with point clouds of robot base leveraging learning-based 3d vision," 2023.

[14] O. Bahadir, J. P. Siebert, and G. Aragon-Camarasa, "A deep learning-based hand-eye calibration approach using a single reference point on a robot manipulator," in *ROBIO*. IEEE, 2022, pp. 1109–1114.

[15] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox, and S. Birchfield, "Camera-to-robot pose estimation from a single image," in *ICRA*. IEEE, 2020, pp. 9426–9432.

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

[17] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *IJCV*, vol. 81, no. 2, pp. 155–166, 2009.

[18] J. Lu, F. Liu, C. Girerd, and M. C. Yip, "Image-based pose estimation and shape reconstruction for robot manipulators and soft, continuum robots via differentiable rendering," *ICRA*, 2023.

[19] L. Chen, Y. Qin, X. Zhou, and H. Su, "Easyhec: Accurate and automatic hand-eye calibration via differentiable rendering and space exploration," *RA-L*, vol. 8, no. 11, pp. 7234–7241, 2023.

[20] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[21] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[22] H. Zhuang, Z. Roth, and R. Sudhakar, "Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form ax=yb," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 4, pp. 549–554, 1994.

[23] X. Zhang, Y. Xi, Z. Huang, L. Zheng, H. Huang, Y. Xiong, and K. Xu, "Active hand-eye calibration via online accuracy-driven next-best-view selection," *The Visual Computer*, vol. 39, no. 1, pp. 381–391, 2023.

[24] J. Yang, J. Rebello, and S. L. Waslander, "Next-best-view selection for robot eye-in-hand calibration," *arXiv preprint arXiv:2303.06766*, 2023.

[25] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *CVPR*, 2016, pp. 4104–4113.

[26] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 224–236.

[27] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *CVPR*, 2020, pp. 4938–4947.

[28] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *CVPR*, 2019.

[29] J. Sun, Z. Wang, S. Zhang, X. He, H. Zhao, G. Zhang, and X. Zhou, "Onepose: One-shot object pose estimation without cad models," in *CVPR*, 2022, pp. 6825–6834.

[30] X. He, J. Sun, Y. Wang, D. Huang, H. Bao, and X. Zhou, "Onepose++: Keypoint-free one-shot object pose estimation without cad models," *NeurIPS*, vol. 35, pp. 35 103–35 115, 2022.

[31] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "Loftr: Detector-free local feature matching with transformers," in *CVPR*, 2021, pp. 8922–8931.

[32] V. Panek, Z. Kukelova, and T. Sattler, "Meshloc: Mesh-based visual localization," in *ECCV*. Springer, 2022, pp. 589–609.

[33] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, 2017, pp. 2961–2969.

[34] A. Kirillov, Y. Wu, K. He, and R. Girshick, "Pointrend: Image segmentation as rendering," in *CVPR*, 2020, pp. 9799–9808.

[35] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight sam for mobile applications," *arXiv preprint arXiv:2306.14289*, 2023.

[36] L. Ke, M. Ye, M. Danelljan, Y. Liu, Y.-W. Tai, C.-K. Tang, and F. Yu, "Segment anything in high quality," *arXiv:2306.01567*, 2023.

[37] F. Li, H. Zhang, P. Sun, X. Zou, S. Liu, J. Yang, C. Li, L. Zhang, and J. Gao, "Semantic-sam: Segment and recognize anything at any granularity," *arXiv preprint arXiv:2307.04767*, 2023.

[38] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[39] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *CVPR*, 2016, pp. 5297–5307.

[40] J. Edstedt, I. Athanasiadis, M. Wadenbäck, and M. Felsberg, "DKM: Dense kernelized feature matching for geometry estimation," in *CVPR*, 2023.

[41] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, L. Yi, A. X. Chang, L. J. Guibas, and H. Su, "SAPIEN: A simulated part-based interactive environment," in *CVPR*, June 2020.

[42] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *CVPR*, 2019, pp. 4561–4570.

[43] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *T-RO*, 2023.

[44] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.