Multiplanar Self-Calibration for Mobile Cobot 3D Object Manipulation using 2D Detectors and Depth Estimation

Tuan Dang Khang Nguyen Manfred Huber

***Abstract*— Calibration is the first and foremost step in dealing with sensor displacement errors that can appear during extended operation and off-time periods to enable robot object manipulation with precision. In this paper, we present a novel multiplanar self-calibration between the camera system and the robot's end-effector for 3D object manipulation. Our approach first takes the robot end-effector as ground truth to calibrate the camera's position and orientation while the robot arm moves the object in multiple planes in 3D space, and a 2D state-of-the-art vision detector identifies the object's center in the image coordinates system. The transformation between world coordinates and image coordinates is then computed using 2D pixels from the detector and 3D known points obtained by robot kinematics. Next, an integrated stereo-vision system estimates the distance between the camera and the object, resulting in 3D object localization. We test our proposed method on the Baxter robot with two 7-DOF arms and a 2D detector that can run in real time on an onboard GPU. After self-calibrating, our robot can localize objects in 3D using an RGB camera and depth image. The source code is available at `https://github.com/tuantdang/calib_cobot`.**

I. Introduction

Precisely manipulating objects in three-dimensional (3D) space is essential in robotic applications and relies heavily on the camera calibration process, including camera position and orientation calibration. As the robot initializes, it needs to know the relative pose between the sensor and its end-effector to localize objects correctly in 3D and thus identify its relative transformation to that object. However, displacements between sensors and end-effectors can occur due to mechanical vibration during the operation, sensor deterioration, or human factors within the working environment during its off-operation time. Thus, the robot eventually loses its knowledge about the relative sensor transformations. For this reason, a multiplanar self-calibration process for mobile cobots is a crucial part of manipulating objects in 3D precisely and efficiently (*e.g.*, pick-and-place task).

Previous works mostly focus on single planar calibration [1], [2], where the camera intrinsic and extrinsic matrices are calibrated using a checkerboard as ground truth in the calibration process. This technique incurs unexpected errors due to human involvement when measuring the distance between the camera and the end-effector as well as the misalignment of the checkboard with respect to the image plane. Therefore, the captured images might be sheared, leading to correlated errors between cells on the checkerboard. Also, this method only helps the robot to precisely localize objects in a 2D plane, which strictly only enables robot manipulation in a fixed plane instead of in arbitrary planes of 3D space within the reachable range of the robot arms.

All authors are with the Learning and Adaptive Robotics Laboratory, Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX 76013, USA. (emails: tuan.dang@uta.edu, khang.nguyen8@mavs.uta.edu, huber@cse.uta.edu)

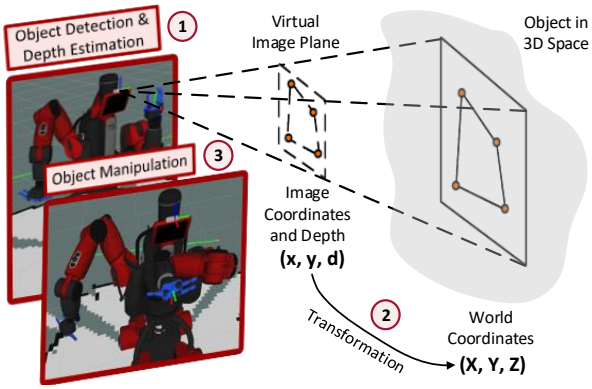


Fig. 1: Multiplanar self-calibration concept.

Recent work proposes multiplanar calibration to manipulate 3D objects by using high-precision machines [3], [4], [5], such as coordinate-milling machines, to obtain the accurate ground truth. However, deploying an external component into the robot system is impractical for robotic applications as cameras could be replaced frequently in response to the needs of the application changes. Thus a re-calibration process for new cameras is needed. Furthermore, the proposed calibration processes are complicated, and require extra equipment, making them impractical for robotic applications.

Self-calibration using proximity sensors [6] provides fully automatic calibration, but this technique suffers from low accuracy due to the lack of spatial information, where only one dimension in space is considered. Other methods estimate the extrinsic matrix without using ground truth [7], [8] by exploring the input data and a probabilistic model. Still, they are impractical and inefficient as they use rich open datasets, requiring the collection of significant amounts of data during calibration.

To fill this gap in camera calibration, we propose a multiplanar self-calibration technique for mobile cobot 3D object manipulation without extra devices (Fig. 1). Our method includes estimating the transformation from camera and robot coordinates to world coordinates and auto-calibrating the scale factor projecting 3D objects onto the image plane. In our calibration process, we only sample two reference planes in 3D, then estimate the 3D points in the range of those two reference planes (in the robot's workspace).

In this work, we make the following contributions: we (1) propose a method to fully auto-calibrate an RGB-D camera on the mobile robot using a state-of-the-art 2D object

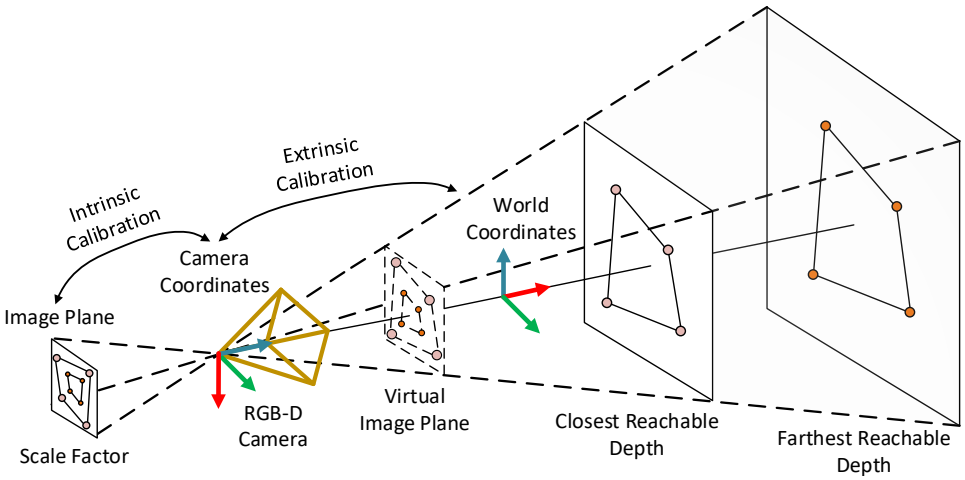


Fig. 2: The calibration process includes intrinsic calibration, extrinsic calibration, and scale factor calibration.

detector without external hardware, (2) build comprehensive software with sample synchronization using the Robot Operating System (ROS) on distributed computers for mobile cobots, and (3) train our 2D detector using two transfer learning strategies and test 3D object manipulation on the Baxter robot with two 7-DOF arms and RGB-D cameras. Our key contribution is to build a fully automated calibration for the robot system. Without using distributed computers and the 2D transfer-learning-based detector, we cannot obtain a fully working robot system. Hence, while focusing on the key contribution, the two last contributions complement our system for task-driven experiments and real-life applications.

II. Related Work

Hand Eye Calibration: Mounting sensors, as well as knowing their relative positions to the robot's rigid body, is a challenging task in robotics. Indeed, the task of determining the pose transformation between sensors and the robot itself is known as the hand-eye calibration problem [9], [10], [7], [11]. In practice, calibration is not only needed after mounting sensors on the robot but also before and after each operation because of unexpected displacements during its operation caused by mechanical vibration or human factors. To estimate the camera pose, researchers concentrate on two main methods, namely the marked and the unmarked method. The marked method utilizes a known pattern as ground truth, in which the camera detects the patterns knowing their world positions [12], [13], [14]. This method is simple and highly accurate as it is purposefully designed in a high-precision setup; however, it requires extensive labor and is not well-suited for real world robot applications due to frequent re-calibration. On the other hand, the unmarked method does not require a known pattern since it infers knowledge from feature extraction and matching. However, this technique requires powerful computation and gives lower accuracy [15], [16]. In this work, we present a technique that achieves accurate calibration results with less overhead (*e.g.,* fewer additional human-involved tasks) and enables the robot to perform pick-and-place tasks in 3D space.

Robot Self-Calibration: Most self-calibration systems need ground truth to estimate the sensor's relative position and orientation, while some do not since they can estimate the ground truth using probabilistic models [7], [8]. Using known patterns like a checkerboard as ground truth is well-known in camera calibration applications [17], [18], [19], while some other work uses extra devices as ground truth, such as optical track [20], proximity [6], or IMU sensors [21], to estimate the extrinsic transformation. In our work, the ground truth is retrieved from the robot hand's relative position each time we sample a 3D-2D point pair, which does not only ensure automation but also independence of taking sample points in each plane (Sec. IV).

Transfer Learning-based 2D Detector & Depth Estimation: 2D detectors using Convolutional Neural Networks (CNNs) have recently been used in many robotic applications due to fast and accurate detection. The two most common approaches for 2D object detection are single-stage detection [22], [23], and two-stage detection [24], [25]. As our design aims to leverage the available hardware to run a real-time 2D detector, we prefer the state-of-the-art single-stage detector, and hence take advantage of the 2D bounding boxes resulting from a 2D detector with an estimated depth from the depth image [26] to reconstruct detected objects' world coordinates. Since this work is extended from [27], we use similar strategies of transfer learning to detect in-lab objects unavailable in the common datasets.

III. Problem Formalization

A. Camera Extrinsic and Scale Factor Calibration

In our work, estimating objects' poses in 3D space based on their pixel coordinates on the image plane captured by a pinhole-like camera model is mathematically written as:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = s \cdot \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (1)$$

where (X_w, Y_w, Z_w) represents the coordinates of the object in the world coordinate system (note that Z_w will be re-estimated using the estimated depth), s is the scale factor, r_{ij} and t_k are coefficients of the rotation matrix, $\mathbf{R}$, and translation matrix, $\mathbf{t}$, respectively, (f_x, f_y) indicates the focal length of the camera, (c_x, c_y) is the center coordinates of

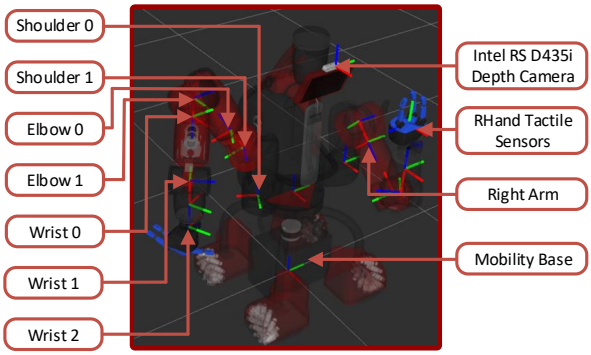


Fig. 3: Robot frames and their kinematic transformations.

the image, and (u, v) represents the pixel coordinates of the object on the image plane. The camera extrinsic matrix can be also written in short form as $[\mathbf{R}|\mathbf{t}]$.

To calibrate the mounted camera on a mobile cobot at a certain depth, we obtain the intrinsic matrix from the preprogrammed camera. Moving to calibration at multiple depths, the scale factor, s, is important in scaling the coordinates to minimize prediction errors. Therefore, we further use projective geometry properties by exploiting the relationship between s_i of plane p_i at depth i and s_j of plane p_j at depth j (Fig. 2). The relationship between the scale factor, s, at a certain depth, d, is hypothetically modeled as:

$$s = f(d) = b_0 + b_1 \cdot d \tag{2}$$

We will further explain the camera extrinsic calibration process and the relationship between scale factor and depth, justify the minimum number of reference planes, and show their involvement in the calibration process in Sec. IV-E.

B. 2D Object Detector

The two most popular approaches in 2D object detection are single-stage and two-stage object detection. The single-stage detection is faster in inference time but underperforms two-stage detection in terms of precision. Similar to recent attempts in building frameworks that support CNNs on low-end computing devices, we utilize OpenVINO to run our models on an onboard GPU. In our calibration method, we utilize single-stage object detection since our design aims to maintain real-time performance on the available hardware.

IV. Implementation

A. Transfer Learning for Object Detection

Most object detectors use rich datasets to train the detectors to obtain high accuracy. This, however, limits their use to objects within their datasets, potentially excluding task-specific objects. If we collect a custom in-lab dataset and train our own detector, the model may be over-fitted since it cannot learn from a sufficiently rich feature domain. To solve this problem, we select transfer learning strategies that allow us to transfer feature knowledge from a model trained with rich datasets such as MS COCO and PASCAL VOC. Here, we model our transfer learning and address two questions: (1) what to transfer from well-trained models to our custom model, and (2) how to transfer that knowledge.

We define a domain as $\mathcal{D} = \{\mathcal{X}, P(X)\}$ where $\mathcal{X}$ represents the feature space, $X = \{x_1, x_2, ...\} \in \mathcal{X}$, and $P(X)$ is its marginal distribution. Let $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$ be the learning task that learns from training pairs (x_i, y_i) with $y_i \in \mathcal{Y}$ in the label space. The objective of transfer learning is to improve the predictive function $P(Y_t|X_t)$ in the target domain $\mathcal{D}_t = \{\mathcal{X}_t, P(X_t)\}$ using knowledge from the source domain $\mathcal{D}_s = \{\mathcal{X}_s, P(X_s)\}$ and source learning task $\mathcal{T}_s = \{\mathcal{Y}_s, P(Y_s|X_s)\}$. Let $P(Y|X) = f(X, \beta)$ where f is the task function. The minimizer for the trainable parameters, β, is written in terms of the loss function, $L(\cdot, \cdot)$, and the task function, f, as follows:

$$\text{argmin}_\beta \sum_X L\left[f\left(X, \beta\right), Y\right] \tag{3}$$

To formally apply the transfer learning definition to the ML domain, we divide the task function into two components: feature extraction (backbone) and detection (head), such that $f(X, \beta) = (f^D \circ f^F)(X, \beta^D, \beta^F)$ where f^D, f^F are task functions for detection and feature extraction, and β^D, β^F are parameters for detection and feature extraction, respectively. The analogous minimizer for β_t^F and β_t^D is:

$$\text{argmin}_{\{\beta_t^D, \beta_t^F\}} = \sum_{X_t} L\left[\left(f_t^D \circ f_t^F\right)\left(X_t, \beta_t^D, \beta_t^F\right), Y_t\right] \tag{4}$$

Since features in the source domain are more generalized and sufficiently cover our target domain, we assume that the feature space in the source domain and target domain are similar. However, our target labels are different ($\mathcal{Y}_s \neq \mathcal{Y}_t$) since we retrain the models with in-lab objects (cone, cube, and sphere). Here we utilize two transfer learning strategies: (1) instance transfer, where the marginal distribution of source features is different from that of target features, and (2) feature representation transfer, where we fit the source feature domain into the target feature domain.

B. Robot Transformation

The robot is made of multiple rigid frames, and two neighbor frames are connected by a joint. We can identify the relative pose between two frames by rotation and transition matrices. To determine a frame pose referenced to the base frame, we use the chain of transformations as follows:

$$T_{0,n} = \prod_{j=0}^{n-1} T_{j,j+1} = T_{0,1} \cdot T_{1,2} \cdot ... \cdot T_{n-1,n} \tag{5}$$

where $T_{0,n}$ is the transformation between the base frame and the target frame and $T_{i,i+1}$ is the transformation between the reference frame i and the neighboring target frame $i+1$.

Each transformation contains rotation and transformation information with respect to the reference frame. In ROS, links between frames are represented in a tree structure, which allows us to easily find the chain of transformations between any two frames by finding a path between two frames within the tree. This implies that we can calculate the pose of the target frame with the known reference's pose. We calculate the hand's pose with reference to the base when generating the 3D ground truth point, as shown in Fig. 3, as $w_h = T_{b,h} \cdot w_b$, where w_h and w_b are the world coordinates of the hand and the base, respectively, and $T_{b,h}$ is the transformation between the base and the robot hand.

C. Homogeneous Points Avoidance

Multiple points on a single ray in space can represent the same pixel on the image plane. To avoid collecting the same point representing the same pixel at multiple depths, we re-validate each acquired Euclidean coordinate by using point and line representations in homogeneous coordinates [28]:

$$\mathfrak{l} = \mathfrak{o} \wedge \mathfrak{p} : \; l = O \times P \tag{6}$$

where $\mathfrak{l}$, $\mathfrak{o}$, and $\mathfrak{p}$ represent a line, the origin coordinates, and an arbitrary point in homogeneous coordinates, respectively, and l, O, and P represent a line, the origin coordinates, and an arbitrary point in the Euclidean coordinate system.

We check if the new point P' lies on line $\mathfrak{l}$ as follows:

$$\mathfrak{p}' \cap \mathfrak{l} : \; P' \cdot \; l = 0 \tag{7}$$

If the dot product results in 0, we discard the new point P'; otherwise, we include P' into our calibration procedure.

D. Data Acquisition Synchronization

Since the robot allows various kinds of sensors to acquire reliable data for perception and navigation tasks, using a single computer would cause less expandability for new sensors and become the bottleneck for tasks that require extensive computing power, such as arm motion planning, navigation, and vision-based tasks. Therefore, we implement our system using distributed computers, where each computer performs a specific task and is synchronized by ROS messages. Specifically, we use one computer to control two robot arms. This computer specializes in low-level motion planning from commands, transforming the end-effector and camera to obtain 3D coordinates and broadcasting them into the ROS network. The second computer detects the target object, gets the bounding box center's image coordinates, calculates the detected object's estimated depth on an input RGB-D image, and broadcasts them to the ROS network. Note that 3D and 2D point coordinates are broadcasted from the two computers at different rates due to their different computing power. Meanwhile, the third computer sequentially sends each calibration point to the first computer and listens to 3D world coordinates and 2D image coordinates from the two other computers. As the arm motion stops, the time synchronization takes place at $T_{sync} = t_{motion} + \min\{\mid t_{motion} - T_{3d} \mid, \mid t_{motion} - T_{2d} \mid\}$ and we sample 3D world and 2D image coordinates. Since motion planning and detection tasks require intensive computing power, no other tasks can interrupt them during their operation. By ensuring this, this design firmly guarantees the real-time performance of the entire robot system, as illustrated in Alg. 1.

Algorithm 1: Sample data with time synchronization

Input : $P := \left[\mathbf{p}_1^w, \mathbf{p}_2^w, ..., \mathbf{p}_{n-1}^w, \mathbf{p}_n^w\right]$
Output: $S := [(\mathbf{p}_1^w, \mathbf{u}_1^c), (\mathbf{p}_2^w, \mathbf{u}_2^c), ..., (\mathbf{p}_n^w, \mathbf{u}_n^c)]$

```
function sample_data(P)
    S = [ ]
    for p ∈ P do
        send_move_command(p)
        p=listen_3D_coordinate(blocking=False)
        u=listen_2D_coordinate(blocking=False)
        wait_motion_finish(blocking=True)
        calculate_synchronized_time()
        S.add_3D_2D_points_pair(p,u)
    return S
```

E. Extrinsic Camera & Scale Factor Calibration

1) Extrinsic Camera Calibration: To recover the extrinsic matrix from a set of n pairs of 3D points in world coordinates and 2D points in image coordinates, we obtain the Uncalibrated Perspective-n-Point (UPnP) method [29]. The goal of UPnP is to minimize the reprojection error as follows:

$$\underset{f_x, f_y, [\mathbf{R}|\mathbf{t}]}{\text{minimize}} \sum_i \left|\left|[u_i, v_i]^T - g\left([x_i, y_i, z_i]^T\right)\right|\right|^2$$

where $[u_i, v_i]$ represents point coordinates on the image plane and $g(\cdot)$ is the function projecting 3D points, $[x_i, y_i, z_i]^T$, in world coordinates to image coordinates. Note that we obtained (f_x, f_y) from our depth camera (Sec. III-A).

In our proposed calibration method, the number of collected points, n, in each plane varies since the width of the captured region gets narrower as the robot arm moves toward itself and vice versa. Furthermore, to uniquely define and make it easy to estimate, we express each of the 3D points in world coordinates in terms of a weighted sum of control points in barycentric coordinates as $\mathbf{p}_i^w = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^w$, where $\mathbf{p}_i^w$ is a 3D point in world coordinates, $\mathbf{c}_j^w$ represent coordinates of four control points in world coordinates, and α_{ij} are homogeneous barycentric coordinates of the i^{th} 3D point, calculated from that 3D point position with respect to the four control points in world coordinates under the normalization condition that $\sum_{j=1}^n a_{ij} = 1$.

Since barycentric coordinate properties pertain to camera coordinates, we can also convert 3D points in camera coordinates to a weighted sum of control points similar to $\mathbf{p}_i^w$:

$$\mathbf{p}_i^c = \sum_{j=1}^4 \alpha_{ij} \mathbf{c}_j^c = \sum_{j=1}^4 \alpha_{ij} \left[x_j^c, y_j^c, z_j^c\right]^T \tag{8}$$

where $\mathbf{p}_i^c$ is a 3D point in camera coordinates and $\mathbf{c}_j^c$ represent coordinates of four control points in camera coordinates.

Substituting Eq. 8 into the inverse of Eq. 1, we obtain:

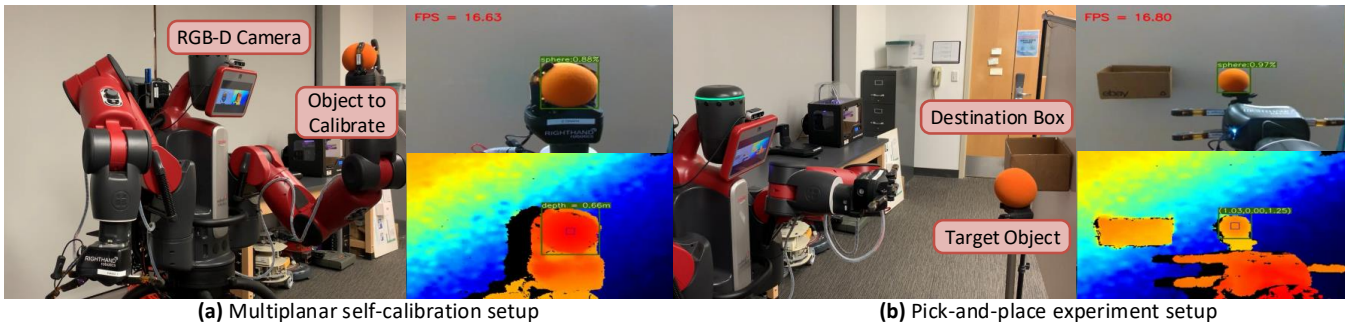
$$s \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left[\mathbf{R} \begin{bmatrix} X_w & Y_w & Z_w \end{bmatrix} + \mathbf{t}\right] =$$

$$= \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}_i^c = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^4 \alpha_{ij} \begin{bmatrix} x_j^c \\ y_j^c \\ z_j^c \end{bmatrix} \tag{9}$$

Leveraging the fact that $s = \sum_{j=1}^4 \alpha_{ij} z_j^c$ from the third row of Eq. 9, we substitute $s = \sum_{j=1}^4 \alpha_{ij} z_j^c$ into the two other rows of Eq. 9 and get two corresponding equations:

$$\begin{cases} \sum_{j=1}^4 \alpha_{ij} x_j^c + \alpha_{ij}(c_x - u_i) \cdot (z_j^c / f_x) = 0 \\ \sum_{j=1}^4 \alpha_{ij} y_j^c + \alpha_{ij}(c_y - v_i) \cdot (z_j^c / f_y) = 0 \end{cases} \tag{10}$$

Thus, each 3D point produces two corresponding equations in terms of homogeneous barycentric coordinates, the center coordinates on the image plane, the control points in barycentric coordinates, and the focal length. We thus need to solve $2n$ equations that are derived from n pairs of 3D-2D points along with 12 unknown coordinates, $\left[x_j^c, y_j^c, z_j^c\right]_{j=1,2,3,4}$, of four control points in the camera coordinates. We express $2n$ equations in matrix form as $\mathbf{Mx} = 0$, where $\mathbf{M}$ is a $2n \times 12$ matrix containing coefficients



(a) Multiplanar self-calibration setup **(b)** Pick-and-place experiment setup

Fig. 4: (a) Multiplanar self-calibration setup for Baxter robot using YOLOv8 as a 2D detector and depth estimation from an RGB-D image captured by a mounted Intel RealSense D435i depth camera, and **(b)** pick-and-place experiment requiring the Baxter robot to localize the target object (sphere) in 3D space, pick it up, and then place it into the destination box.

α_{ij} and the coordinates of points, $[u_i, v_i]^T$, on the image plane. Here, $\mathbf{x}$ is obtained using the Exhaustive Linearization and Relinearization method, as described in [29].

2) Scale Factor Calibration: Since the scale factor, s, acts as an intermediate value, it is no longer determined in the calibration process, as described in [30]. In our work, to calibrate the scale factor at each 3D reference plane, we find the scale factor as described in the following equation:

$$\underset{s}{\operatorname{argmin}} \sum_{i=1}^{N} \left[\begin{bmatrix} X_{w,i} \\ Y_{w,i} \\ Z_{w,i} \\ 1 \end{bmatrix} - s \cdot \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} [\mathbf{R}|\mathbf{t}]^{-1} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \right]^2 \quad (11)$$

As illustrated in Fig. 2, we calibrate the scale factors, s_1 and s_2, at the closest reachable depth and the farthest reachable depth, respectively. Here we leverage the projective geometry property to establish the relationship between scale factors of depths that lie in the range of the closest reachable depth and the farthest reachable depth, as specified in Eq. 2.

F. Summary of Calibration Procedure

Our proposed calibration method is summarized as below:

1) Move the object to different known positions on the first plane at depth, d_1, and detect the object locations on the image plane using the 2D detector.
2) Estimate the camera extrinsic matrix, $[\mathbf{R}|\mathbf{t}]$, based on the set of 3D-2D point pairs resulting from Alg. 1, as described in Sec. IV-E.1.
3) Estimate the first scale factor, s_1, for the first plane at the first depth, d_1, using Eq. 11.
4) Repeat Step 1 at depth d_2 to obtain the set of 3D-2D point pairs, and then estimate the second scale factor, s_2, based on the obtained extrinsic matrix $[\mathbf{R}|\mathbf{t}]$ in Step 2 and the newly-obtained set of 3D-2D point pairs.
5) Estimate the scale factor based on the estimated depth: $s = b_0 + b_1 \cdot d$.

V. Experimental Results & Evaluation

A. Experimental Setup

We first mount the Intel RealSense D435i depth camera on the display of the Baxter robot (Fig. 3). Then, we let the robot hold one of the objects (cone, cube, and sphere) in its hand. As a set of 3D points on a plane at a distance of d_1 from the robot is generated, the robot executes the set of 3D points following Alg. 1. Specifically, after the robot executes a position command, the feedback position is read as 3D point ground truth to address that the feedback position may differ slightly from the position command due to the hardware imperfection. At the same time, we use YOLOv8 to detect the bounding box of the object and calculate the center coordinates of the bounding box on the image plane (Fig. 4a). We repeat the process iteratively for n 3D points and use them to estimate the extrinsic matrix (Sec. IV-E.1) and the scale factor s_1 (Sec. IV-E.2). The process is repeated on the second plane at a depth of d_2 (Sec. IV-C), thus obtaining the second scale factor, s_2. For calibration testing, we repeat the process on multiple planes that are lying in the robot workspace. In verifying the correctness of our calibration method, we task the Baxter robot with picking the object and putting it into the destination box (Fig. 4b).

B. Evaluation Metrics

To evaluate how well the model detects, we compute average precision (AP) for each object class and mean average precision (mAP) for all classes at multiple intersections over union (IoU) thresholds and compare the precision among three transfer learning strategies. For the calibration precision, we calculate the error along each axis and the Euclidean distance between ground truth points and estimated points.

1) Object Detection: : We train the YOLOv8-tiny model on the NVIDIA GTX 4090 (24 GB) GPU with three strategies mentioned in Sec. IV-A with 1000 epochs for each strategy. The trained model starts to converge at the 150th epoch and finally converges at the 700th epoch, taking approximately 43 minutes. To assess how well the transfer learning strategies are during the training stage, we calculate AP and mAP based on IOU thresholds based on the ground truth bounding box and the proposal bounding box. The IoU thresholds range from 0.01 to 1.00 with a step of 0.01. After evaluating detection proposals on all IoU thresholds, we calculate the mAP for each model. The results of AP and mAP on IoU thresholds from 0.80 to 1.00 are shown in Fig. 5. Overall, two transfer learning strategies outperform training from scratch, which means the source domain knowledge is useful in target tasks. The feature extraction strategy gives similar results to the fine-tuning strategy, which implies that the features in the source domain cover most of the features in the target domain. Lastly, the feature extraction strategy outperforms the fine-tuning strategy in detecting the sphere.

2) Calibration Error: After calibrating the scale factor based on the estimated depth from the depth camera, we

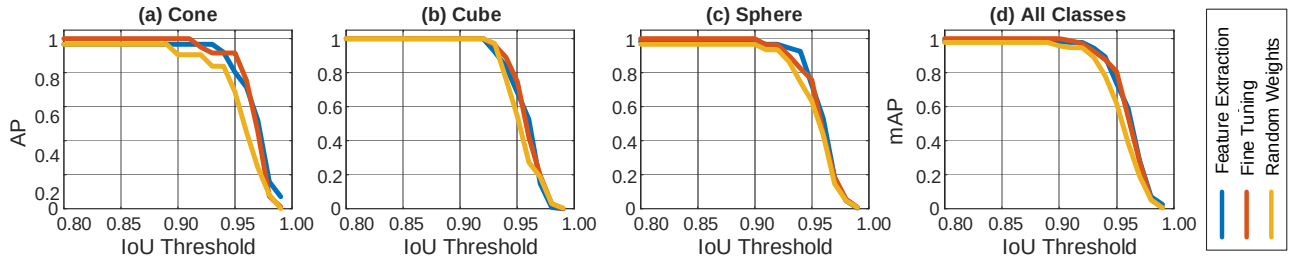


Fig. 5: Average precision (AP) for each class (cone, cube, and sphere) and mean average precision (mAP) among all classes on the YOLOv8-tiny model. The model is trained with three different strategies, as described in Sec. IV-A.

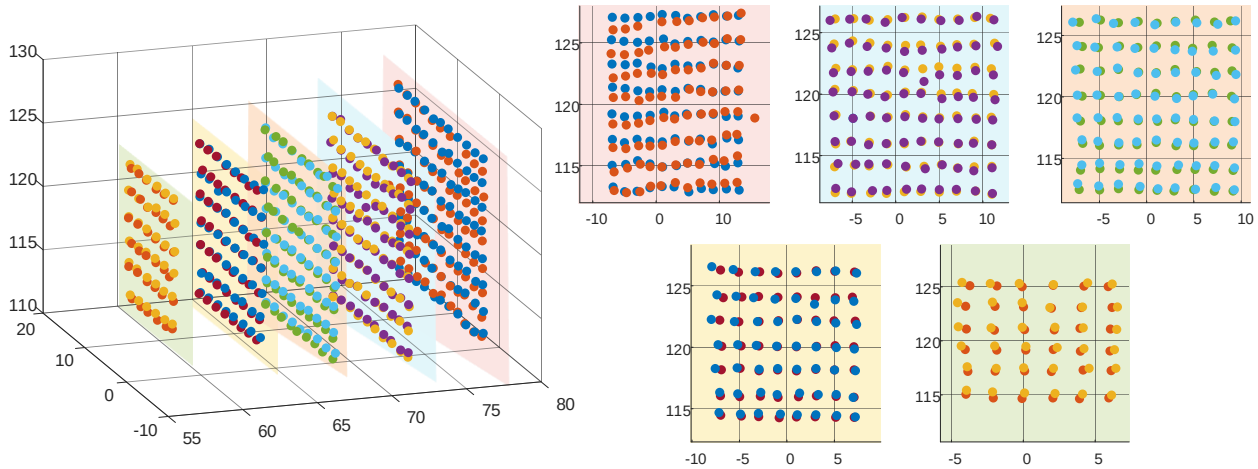


Fig. 6: Results of our proposed calibration method at depths of 0.57, 0.62, 0.67, 0.72, and 0.77 meters.

make multiple predictions for the world coordinates of each detected object position on the image plane. We compare them with ground truth positions obtained from the robot transformation (Sec. IV-B). Here, we calculate the displacement errors along the x-axis, y-axis, and z-axis in world coordinates and the Euclidean distance error between the ground truth positions and the predicted positions. The calibration error results are presented in Table I. For the x-axis, the maximum error occurs at a distance of 0.77 m, and the minimum error at 0.57 m. For the y-axis, the maximum error is at 0.77 m, and the minimum error at 0.72 m. For the z-axis, the maximum error is at 0.62 m, and the minimum error at 0.77 m. For the Euclidean distance error, the maximum error occurs at 0.77 m, and the minimum error occurs at 0.72 m. Fig. 6 shows the resulting 3D points are reconstructed from 2D points, and no systematic error is found as each sample position is independent of the other. Since our robot moves the object in 3D and holds it in the air, small vibrations are inevitable, while other static systems, such as turntables or coordinate-milling machines, do not encounter such issues. However, our method induces Euclidean distance errors ranging from 3.9 mm to 6.5 mm on multiple plans within the robot's workspace, which outperforms the average error of 6.6 mm with the variance of 6.0 mm on a pre-defined plane reported from a recently-proposed method [6].

C. Demonstration

The demonstration video includes three parts: (1) the robot performing multiplanar self-calibration using YOLOv8 as a 2D detector and depth estimation simultaneously, (2) the robot testing 3D estimation using an RGB-D image from the Intel RealSense D435i depth camera, and (3) the robot picking a detected target object (sphere) and then placing it in the destination box: `https://youtu.be/KrDJ22rvOAo`.

Depth	Mean Error	Avg Error (x)	Avg Error (y)	Avg Error (z)
0.57 m	0.0039	0.0026	0.0022	0.0007
0.62 m	0.0030	0.0014	0.0020	0.0008
0.67 m	0.0032	0.0019	0.0018	0.0007
0.72 m	0.0028	0.0019	0.0015	0.0005
0.77 m	0.0065	0.0047	0.0033	0.0004

TABLE I: Calibration errors, including mean Euclidean distance error, average error on the x-axis, y-axis, and z-axis, calculated in meters at depths of 0.57, 0.62, 0.67, 0.72, and 0.77 meters.

VI. Conclusion & Future Works

This work proposes a novel multiplanar self-calibration method for mobile cobots utilizing a 2D object detector with depth estimation from RGB-D images acquired from the Intel RealSense D435i depth camera. The robot self-calibrates using its end-effector as ground truth for the camera's position and orientation. Meanwhile, the 2D detector allows the robot to identify the object's proposal prediction bounding box, locate the center coordinates of the bounding box, and estimate the depth of the detected object. Through this, the robot calculates the transformation from the image coordinates and the depth of the object in world coordinates. We tested our proposed self-calibration method on a two 7-DOF arm Baxter robot. After calibrating and calculating the world coordinates of the detected-target object, the robot is able to perform pick-and-place tasks in 3D precisely and efficiently. We reserve the object manipulation task based on 3D object segmentation, classification, and detection after calibrating the robot's camera for future work.

REFERENCES

[1] M. Li, Z. Du, X. Ma, W. Dong, and Y. Gao, "A robot hand-eye calibration method of line laser sensor based on 3d reconstruction," *Robotics and Computer-Integrated Manufacturing*, vol. 71, p. 102136, 2021.

[2] C. Yuan, X. Liu, X. Hong, and F. Zhang, "Pixel-Level Extrinsic Self Calibration of High Resolution LiDAR and Camera in Targetless Environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7517–7524, Oct. 2021, conference Name: IEEE Robotics and Automation Letters.

[3] J. M. S. T. Motta, G. C. de Carvalho, and R. S. McMaster, "Robot calibration using a 3D vision-based measurement system with a single camera," *Robotics and Computer-Integrated Manufacturing*, vol. 17, no. 6, pp. 487–497, Dec. 2001.

[4] M. Gaudreault, A. Joubair, and I. Bonev, "Self-Calibration of an Industrial Robot Using a Novel Affordable 3D Measuring Device," *Sensors*, vol. 18, no. 10, p. 3380, Oct. 2018, number: 10 Publisher: Multidisciplinary Digital Publishing Institute.

[5] G. Ma, W. Ross, and P. J. Codd, "StereoCNC: A Stereovision-guided Robotic Laser System," Sept. 2021, pp. 540–547, iSSN: 2153-0866.

[6] K. Watanabe, M. Strong, M. West, C. Escobedo, A. Aramburu, K. C. Kodur, and A. Roncone, "Self-Contained Kinematic Calibration of a Novel Whole-Body Artificial Skin for Human-Robot Collaboration," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2021, pp. 1778–1785, iSSN: 2153-0866.

[7] H. Li, Q. Ma, T. Wang, and G. S. Chirikjian, "Simultaneous Hand-Eye and Robot-World Calibration by Solving the $AX=YB$ Problem Without Correspondence," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 145–152, Jan. 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7349161/

[8] X. Hu, D. Olesen, and K. Per, "A Novel Robust Approach for Correspondence-Free Extrinsic Calibration," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 1–6, iSSN: 2153-0866.

[9] R. Horaud and F. Dornaika, "Hand-Eye Calibration," *The International Journal of Robotics Research*, vol. 14, no. 3, pp. 195–210, June 1995, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/027836499501400301

[10] K. H. Strobl and G. Hirzinger, "Optimal Hand-Eye Calibration," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 4647–4653, iSSN: 2153-0866.

[11] C. Fang, S. Ding, Z. Dong, H. Li, S. Zhu, and P. Tan, "Single-shot is enough: Panoramic infrastructure based calibration of multiple cameras and 3d lidars," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8890–8897.

[12] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000, conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[13] C. Celozzi, G. Paravati, A. Sanna, and F. Lamberti, "A 6-DOF ARTag-based tracking system," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 1, pp. 203–210, Feb. 2010, conference Name: IEEE Transactions on Consumer Electronics.

[14] S. M. Abbas, S. Aslam, K. Berns, and A. Muhammad, "Analysis and Improvements in AprilTag Based State Estimation," *Sensors*, vol. 19, no. 24, p. 5480, Jan. 2019, number: 24 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/1424-8220/19/24/5480

[15] L. Hua, W. Fu-Chao, and H. Zhan-Yi, "A new linear camera self-calibration technique," *Chinese J. Computers*, vol. 23, no. 11, pp. 1121–1129, 2000.

[16] S. J. Maybank and O. D. Faugeras, "A theory of self-calibration of a moving camera," *International journal of computer vision*, vol. 8, no. 2, pp. 123–151, 1992.

[17] X. Zhi, J. Hou, Y. Lu, L. Kneip, and S. Schwertfeger, "Multical: Spatiotemporal Calibration for Multiple IMUs, Cameras and LiDARs," Oct. 2022, pp. 2446–2453, iSSN: 2153-0866.

[18] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3936–3943.

[19] A.-I. García-Moreno, J.-J. Gonzalez-Barbosa, F.-J. Ornelas-Rodriguez, J. B. Hurtado-Ramos, and M.-N. Primo-Fuentes, "LIDAR and Panoramic Camera Extrinsic Calibration Approach Using a Pattern Plane," in *Pattern Recognition*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, J. S. Rodríguez, and G. S. di Baja, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, vol. 7914, pp. 104–113, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-642-38989-4_11

[20] C. Wang, J. Cartucho, D. Elson, A. Darzi, and S. Giannarou, "Towards Autonomous Control of Surgical Instruments using Adaptive-Fusion Tracking and Robot Self-Calibration," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 2395–2401, iSSN: 2153-0866.

[21] C. Walters, O. Mendez, S. Hadfield, and R. Bowden, "A Robust Extrinsic Calibration Framework for Vehicles with Unscaled Sensors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 36–42, iSSN: 2153-0866.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Las Vegas, NV, USA, June 2016, pp. 779–788. [Online]. Available: http://ieeexplore.ieee.org/document/7780460/

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[26] F. Tosi, F. Aleotti, M. Poggi, and S. Mattoccia, "Learning Monocular Depth Estimation Infusing Traditional Stereo Knowledge," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 9791–9801. [Online]. Available: https://ieeexplore.ieee.org/document/8954201/

[27] T. Dang, K. Nguyen, and M. Huber, "Perfc: An efficient 2d and 3d perception software-hardware framework for mobile cobot," in *The International FLAIRS Conference Proceedings*, vol. 36, 2023.

[28] R. F. Riesenfeld, "Homogeneous coordinates and projective planes in computer graphics," *IEEE Computer Graphics and Applications*, vol. 1, no. 01, pp. 50–55, 1981.

[29] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer, "Exhaustive linearization for robust camera pose and focal length estimation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 10, pp. 2387–2400, 2013.

[30] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Ep n p: An accurate o (n) solution to the p n p problem," *International journal of computer vision*, vol. 81, pp. 155–166, 2009.