VOLUME 19, N° 3 2025
Journal of Automation, Mobile Robotics and Intelligent Systems

Monocular 3D Object Localization using 2D Estimates for Industrial Robot Vision System

Submitted: 13th June 2024; accepted: 13th September 2024

Thanh Nguyen Canh, Du Trinh Ngoc, Xiem HoangVan

DOI: 10.14313/jamris-2025-025

Abstract:
3D Object Localization has emerged as one of the pivotal challenges in Machine Vision tasks. In this paper, we proposed a novel 3D object localization method, leveraging a blend of deep learning techniques primarily rooted in object detection, post-image processing, and pose estimation algorithms. Our approach involves 3D calibration methods tailored for cost-effective industrial robotics systems, requiring only a single 2D image input. Initially, object detection is performed using the You Only Look Once (YOLO) model, followed by an R-CNN model for segmenting the object into two distinct parts, i.e., the top face and the remaining parts. Subsequently, the center of the top face serves as the initial positioning reference, refined through a novel calibration algorithm. Our experimental results indicate a significant enhancement in localization accuracy, showcasing the method's efficacy in reducing localization errors broadly across various testing scenarios. We have also made the code and datasets openly accessible to the public at (https://github.com/NguyenCanhThanh/MonoCalibNet)

Keywords: *Camera Calibration, Object Localization, Machine (Robot) Vision System, Industrial Robotics*

1. Introduction

The rapid advancement of imaging sensors over recent decades has paved the way for a plethora of intelligent perception algorithms [1, 2]. Leveraging these capabilities, vision technology has made significant strides in various fields, including space robotics [3], robot manufacturing, rapid object detection, and tracking. Industrial Robot Vision (IRV), which integrates computer vision into industrial manufacturing processes, presents a nuanced approach compared to traditional computer vision methodologies. Typically, robot vision systems prioritize tasks such as harvesting [4], human-robot interaction [5], and robot navigation [6]. These systems find application across a spectrum of areas, including complex system part identification, defect inspection, Optical Character Recognition (OCR) reading, 2D code reading, piece counting, and dimensional measurement [7]. Figure 1 illustrates a typical industrial robot vision system configuration, comprising fundamental components such as cameras and control systems (*e.g.*, Robots, PLCs). Additional features like illumination, user interaction, data storage, and remote control are gradually integrated to improve system efficiency. Object images captured are typically subjected to preprocessing, segmentation, and feature extraction on a server. Controlled lighting conditions and fixed camera positions ensure the prominence of critical features, while the control system receives task execution instructions from the server.

Camera calibration is indispensable in robot vision systems to ensure precise object location and accurate measurements [8]. However, the process is highly susceptible to environmental changes, encompassing variations in lighting, temperature, and humidity, potentially introducing inaccuracies by impacting intrinsic and extrinsic parameters. Despite efforts to precisely estimate camera parameters, real-world complexities such as lens distortions and non-linearities may not be fully accounted for, leading to calibration inaccuracies. Notably, challenges arise when the object is not directly under the camera, resulting in an incorrect prediction of the object's center relative to the reference point. To mitigate these challenges, two primary approaches are considered: i) optimization of intrinsic parameters and ii) recognition of 3D objects to estimate the center point. Numerous algorithms have been developed to obtain intrinsic parameters for imaging sensors [9–11]. Intrinsic calibration typically involves a camera that observes anchor points in a calibration pattern, with commonly used patterns including checkerboards [12], coplanar circles [13, 14], and AprilTags [15]. Traditional calibration methods, such as those supported by OpenCV [16] and MATLAB [17], have become commonplace, leveraging the advantage of calibration toolboxes. However, deploying these approaches in factory settings proves challenging, as they are susceptible to noise and artifacts that can degrade calibration performance.

For industrial vision systems, several techniques for 2D camera calibration have been introduced to achieve precise object localization. Brown's Plumb Line method [18], an early notable approach, addresses the lens asymmetry issues encountered during manufacturing. This method requires the determination of 10 parameters and achieves high accuracy, with deviations as low as $\pm 0.5mm$ at a distance of $2m$.

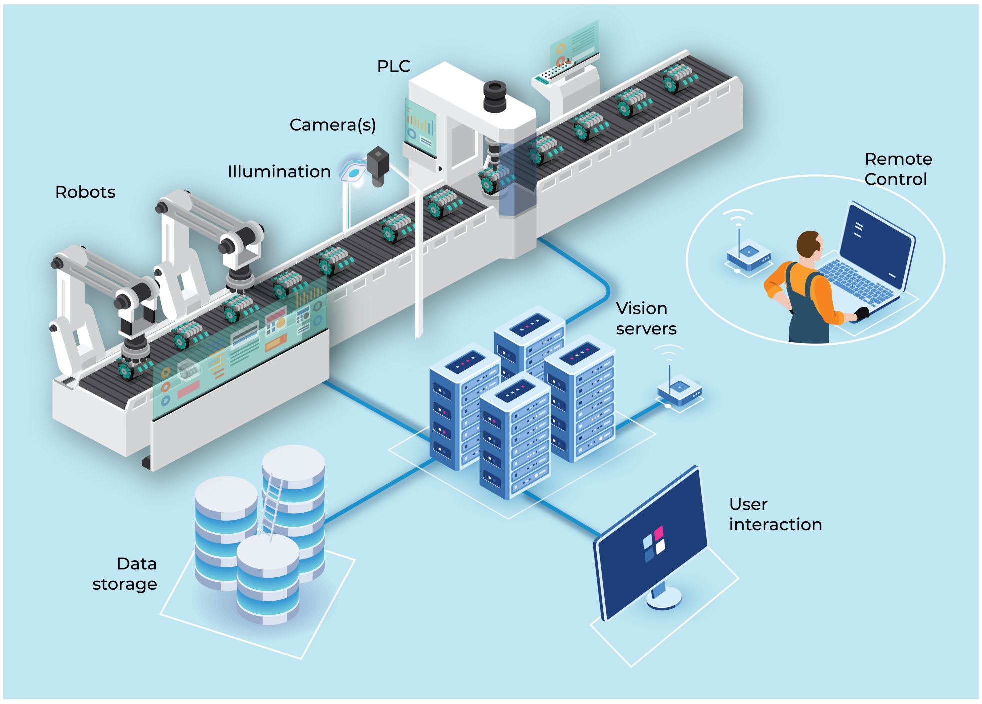


Figure 1. Overview of the industrial robot vision system

Innovations by Clarke, Fryer and Chen [19] have adapted this method to use with CCD sensors, enhancing efficiency. Lu and Chuang [20] utilized a flat monitor on which they drew lines and then estimated the projection between the image plane and the monitor plane through multiple shots to calibrate the camera. The Two-Stage method [21–23] focuses on real-time calibration using black squares on a white background, achieving uncertainties around $\pm 1mm$ at $2m$. Direct Linear Transformation (DLT)-based methods [24–27] simplify calibration models and have been widely adopted. However, these approaches tend to fail in the case of localizing 3D-shaped objects. Due to their 3D natural shape, the locations determined by these calibration methods are the locations of their projections on the image rather than their true spatial positions. Consequently, the error distance to the real location of the objects remains significant. Zhang's technique [28], requiring only a planar pattern, offers flexibility with quicker application in industrial contexts, albeit with slightly higher uncertainties. Beyond these, researchers have tailored calibration methods for specific applications, such as high-speed tensile testing machines and unmanned vehicle guidance, providing unique insights and experimental results. However, these approaches tend to fail in the case of localizing 3D-shaped objects. Building upon foundational studies such as Siddique et al. [29], who explored 3D object localization using 2D estimates for computer vision applications, our work seeks to advance these concepts by integrating more sophisticated calibration methods and deep learning techniques for improved accuracy and efficiency in industrial settings.

Due to their 3D natural shape, the location determined by calibration methods is in fact the location of their projections on the image instead of their real location. Therefore, the error distance to the real location of the objects still remains. In addition, Xiem HoangVan and Nam Do [30] introduced a machine learning – regression-based method for improving the accuracy of 3D object localization. Our method is created based on mathematical modeling of 3D objects and their projected image in the 2D plane and is followed by a regression-based algorithm to achieve model parameters.

In this paper, we proposed a novel approach (M-Calib) with the following contributions:

1) We validated the proposed work in rigorous experiments using a checkerboard. The results show that our approach outperforms the previous in estimation accuracy.
2) We propose an efficient 3D localization method designed to accurately calculate the translation vector between the calibration center target and the initialized center point with sub-pixel localization accuracy. This method demonstrates robustness to noise, ensuring reliable performance in various conditions.
3) We provide the source code of M-Calib to the research community, offering an easy-to-use calibration toolbox specifically tailored for monocular cameras. This resource is openly accessible at: https://github.com/NguyenCanhThanh/MonoCalibNet, facilitating further research and application development in the field.

The remainder of this paper is organized as follows: Section 2 delves into the intricacies of the problem statement, offering a comprehensive understanding of the challenges at hand.

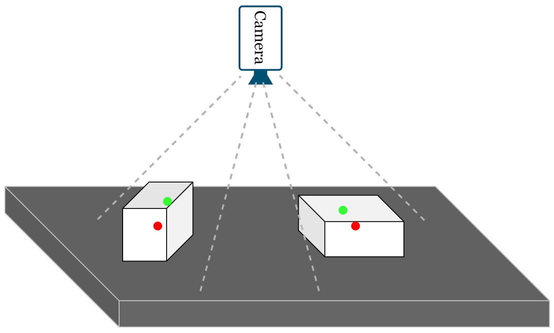


Figure 2. Illustration of industrial robot vision system: the green point is the initialized estimate center point, and the red point is the actual center point

We unveil our novel method for isometric flat 3D object localization, elucidating the deep learning methodologies employed and elucidating the procedural steps in Section 3. Section 4 presents the experimental setup and evaluates the performance of our proposed method using relevant metrics. Finally, Section 5 concludes the paper, summarizing the key findings, discussing potential future research directions, and emphasizing the significance of our contributions.

2. Problem Statement

The problem addressed in this study lies at the intersection of machine vision and 3D object localization within industrial robot vision systems. While conventional 2D camera calibration methods, such as Brown's Plumb Line Method and Tsai's Two-Stage Method, have proven effective for achieving high accuracy in object localization, their limitations become evident when confronted with 3D-shaped objects. The inherent challenge arises from the fact that these methods determine object locations based on their projections onto the 2D image plane, resulting in inaccuracies in representing the true 3D positions. This discrepancy is especially pronounced in industrial contexts where precise object localization is crucial for tasks such as robotic automation, and quality control. Figure 2 illustrates the challenges when estimating the center of 3D objects. Under the influence of optical projection, the initial estimated center position (green point) often tends to deviate from the actual center (red point) position. To address this gap, we propose a novel approach, M-Calib, leveraging efficient 3D localization techniques to overcome the limitations of traditional 2D calibration methods. The objective is to enhance accuracy, particularly in the localization of isometric flat 3D objects, thereby contributing to the advancement of machine vision applications in industrial environments.

3. Proposed Method

Figure 3 presents a visual representation of our proposed calibration methodology, centered around the deliberate choice of a checkerboard as the calibration pattern for two significant reasons.

Firstly, the checkerboard pattern demonstrates robustness against scenes that are out of focus [31]. Secondly, the features extracted from the checkerboard offer a straightforward definition of the original coordinate, contrasting with asymmetric circle patterns, where features are more suitable for motion determination. The checkerboard pattern plays a pivotal role in precisely determining the object's position in the calibration process. As the monocular cameras sweep the calibration pattern, known as O_r corner, we leverage the You Only Look Once (YOLO) model [32], a state-of-the-art object detection model, to identify objects within the camera's field of view. Once the fixed original coordinate is defined, sub-pixel localization accuracy is crucial for extracting the image centers of the calibration targets to optimize sensor calibration. This process is detailed in Section 3.1. However, owing to the direction of light, the object's position may drift away from the actual center. To address this, we introduce a novel calibration method comprising two parts. Initially, we segment the object into the upper plane (S_u) and the lower plane (S_l), as discussed in Section 3.2, utilizing the Bilateral filtering algorithm to eliminate noise. Subsequently, we determine the center of the upper plane (P_u) and employ an edge detection algorithm to extract the edge (E_o) of the lower part. This edge (E_o) is then divided into two main border lines: the upper line (E_u) and the lower line (E_l). The translation vector ($\mathbf{T}$) from (E_u) to (E_l) is calculated, as detailed in Section 3.3. Finally, the estimated object position is computed by shifting the center of the upper part (P_u) following the translation vector ($\mathbf{T}$) with a magnitude of $1/2$.

3.1. Object Detection

In determining the precise position of the object, we initiate the process by establishing real-world coordinates through the capture of a checkerboard image, enabling the identification of its corners. Subsequently, for each object present in the image, we leverage an advanced object detection method to discern their respective image coordinates. In the context of detecting checkerboard corners, we establish the correlation between the image coordinates and their corresponding real-world coordinates. Notably, the relationship between distances in images and their counterparts in the real world is not always linear due to distortion. Figure 4 delineates the sequential steps employed to ascertain the initial object position:

1) Detect Checkerboard Edges and Corners: Utilizing the Hough transformation, we identify checkerboard edges and the coordinates of their intersections, namely, checkerboard corners, as depicted in Figure 4a.
2) Select Fixed Real-World Coordinate: Choose a fixed real-world coordinate in pixel image space, denoted as $O_r = \begin{bmatrix} x_0 & y_0 \end{bmatrix}^T$.
3) Estimate Object Center Position: Employing object detection, estimate the center position in pixel image space, represented as $P = \begin{bmatrix} x & y \end{bmatrix}^T$.

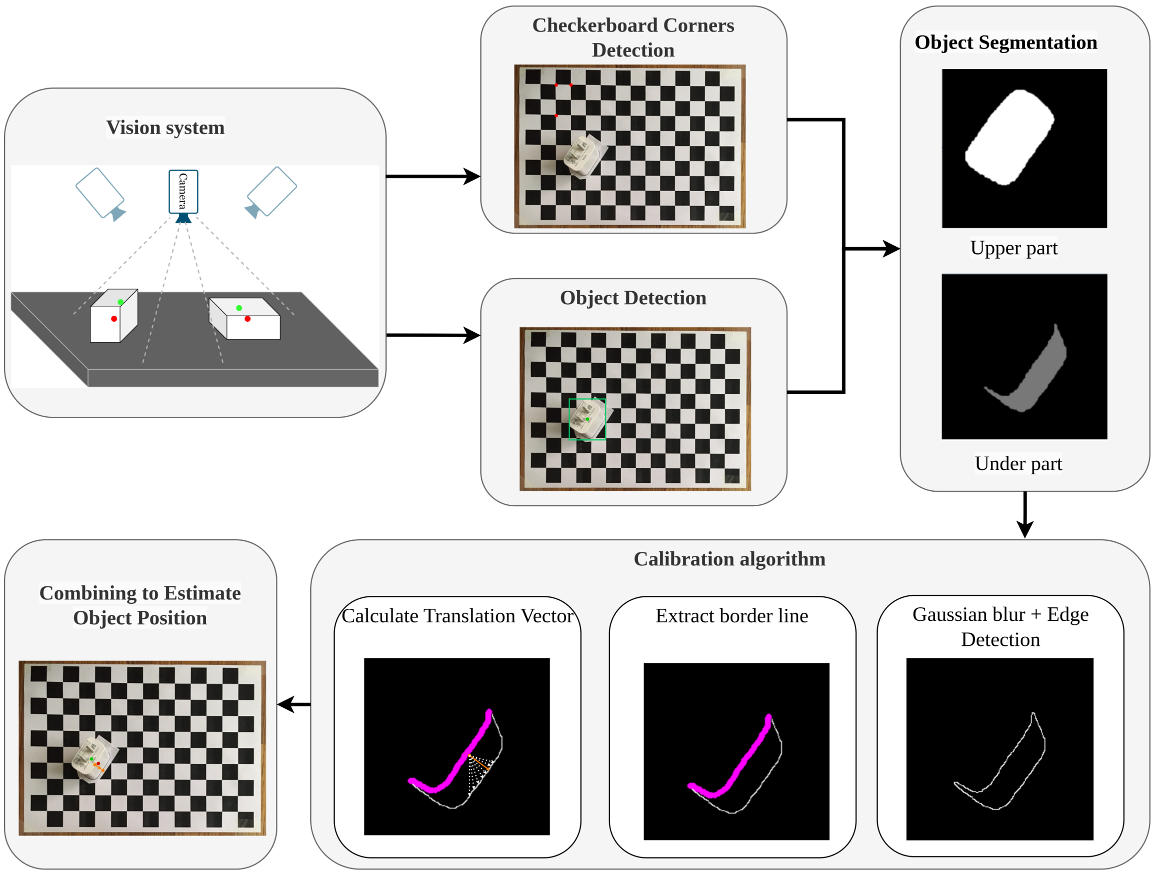
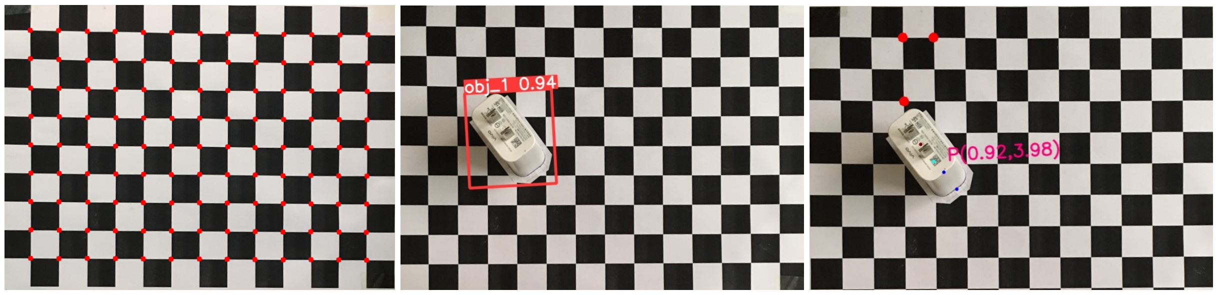

Vision system
Camera
Checkerboard Corners Detection
Object Detection
Object Segmentation
Upper part
Under part
Calibration algorithm
Calculate Translation Vector
Extract border line
Gaussian blur + Edge Detection
Combining to Estimate Object Position

Figure 3. A block diagram of our proposed calibration method. The translation vector between the initialized estimate center point (green point), and the calibration center point (red point) is calculated based on deep learning, and our novel calibration method



(a) Checkerboard detection **(b)** Object detection **(c)** Initial object position

Figure 4. The progress of the calculation of the object position in the real-world coordinate

4) Calculate Object Position in Real-World Coordinates: Compute the object's position in real-world coordinates using the formula $P_r = (P - O_r)r$, where r signifies the average ratio between the length in pixels and the length in the real world of each cell in the checkerboard pattern.

3.2. Object segmentation

While utilizing the checkerboard calibration method to estimate the location of objects proves to be straightforward and readily applicable in industrial settings, its efficacy diminishes significantly when dealing with 3D-shaped objects. In instances involving such objects, the bounding box generated through deep learning methods may not align accurately with the true object location.

Specifically, the coordinates of the bounding box's center are unlikely to correspond to the actual center of the object within the resultant image. It's crucial to note that the bounding box's center can accurately represent the object's center only if the object is precisely positioned at the center of the camera's projection onto the floor. To address this limitation, we employ a convolutional neural network (CNN)-based object segmentation approach to divide objects, detected in Section 3.1, into two distinct planes. This segmentation involves using the upper plane S_u to establish the initial center of the object, followed by noise reduction and edge extraction from the lower plane.

The initial step involves applying a Bilateral filtering operation, which can be expressed mathematically as:

$$I'(p) = \frac{1}{W_p} \sum_{q \in \Omega} I(q) \cdot G_{\sigma_s}(||p-q||) \cdot G_{\sigma_r}(||I(p)-I(q)||) \quad (1)$$

where:
- $I'(p)$ is the filtered intensity at pixel p.
- W_p is the normalization term.
- $I(q)$ is the intensity at pixel q.
- G_{σ_s} is the spatial Gaussian kernel with standard deviation σ_s.
- G_{σ_r} is the range Gaussian kernel with standard deviation σ_r.
- Ω is the spatial neighborhood of the pixel p.

Here, k is the size of the kernel, and σ is the standard deviation of the Gaussian distribution.

After that, the edge detector includes gradient computation, non-maximum suppression, and edge tracking by hysteresis. The magnitude of the gradient (G) and the gradient direction (θ) are calculated as follows:

$$G(i,j) = \sqrt{G_i(i,j)^2 + G_j(i,j)^2} \quad (2)$$

$$\theta(i,j) = arctan(\frac{G_j(i,j)}{G_i(i,j)}) \quad (3)$$

where G_i and G_j are the partial derivatives of the image. After obtaining the gradient magnitude, non-maximum suppression is applied to thin the edges. Finally, edge tracking by hysteresis involves setting two thresholds, T_{high} and T_{low}. Any edge pixel with a gradient magnitude above T_{high} is considered a strong edge, and pixels connected to strong edges and with a magnitude above T_{low} are considered weak edges.

Then we divide the edge into two main border lines: the upper line (E_u) and the lower line (E_l). The upper line is the contact line between the two planes S_u and S_l, and the lower line is the boundary of the lower plane S_l. Figure 5 illustrates the progress of object segmentation and edge extraction.

3.3. Calibration Method

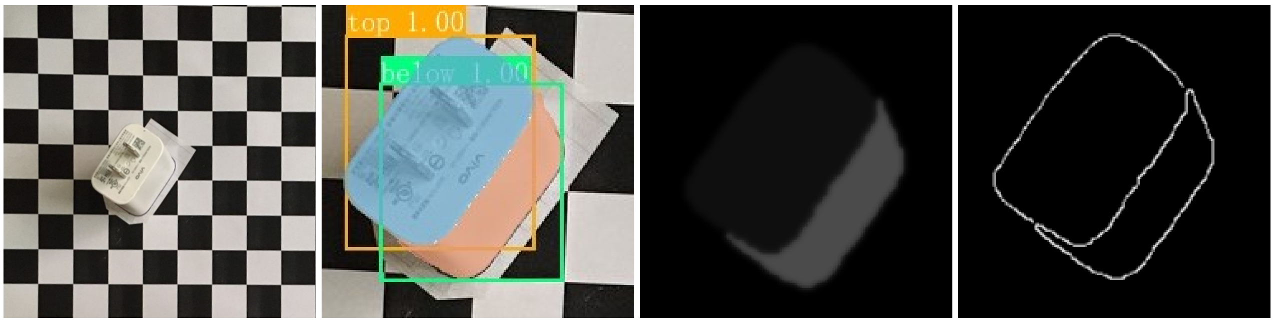
We introduce a calibration technique designed for object position calibration. Upon completion of the object segmentation and edge extraction processes, we obtain crucial components: the initial position of the object (center of the upper plane S_u), the upper line E_u, and the lower line E_l. The translation vector V_{ul} is then calculated using Algorithm 1 to predict the final position of the object by shifting the initial point according to V_{ul}. In this process, each point on the upper line E_u calculates its distance to the lower line S_l, determining the smallest vector length. The visualization results are depicted in Figure 6a. Subsequently, the translation vector V_{ul} is defined as the vector with the maximum length within the set of distance vectors for each point, as illustrated in Figure 6b.

Algorithm 1: Estimate Translation Vector

```
Input: $\mathcal{P}_u = E_{u_1}, E_{u_2}, \dots, E_{u_N}$
       $\mathcal{P}_l = E_{l_1}, E_{l_2}, \dots, E_{l_M}$
       $N$ is the number of points in line $E_u$
       $M$ is the number of points in line $E_l$
Output: Translation vector: $\vec{V_{ul}}$
        Index point in line $E_u$: $ind_u$
        Index point in line $E_l$: $ind_l$
begin:
    $i \leftarrow 0$ ;
    $j \leftarrow 0$ ;
    $ind_u, ind_l \leftarrow 0$ ;
    $\mathcal{D}_{E_u E_l} \leftarrow 0$;    ▷ The magnitude of the translation vector
    while $i < (N-1)$ do
        $\mathcal{D}_{iE_l} \leftarrow +\infty$;    ▷ The distance from a point to a line
        while $j < (M-1)$ do
            Calculate the distance between two points $(\mathcal{P}_{u_i} = (x_{u_i}, y_{u_i}))$ and
            $(\mathcal{P}_{l_j} = (x_{l_j}, y_{l_j}))$ based on (4):
                $\mathcal{D} \leftarrow \sqrt{(x_{u_i} - x_{l_j})^2 + (y_{u_i} - y_{l_j})^2}$    (4)
            if $\mathcal{D}_{iE_l} \geq \mathcal{D}$ then
                $\mathcal{D}_{iE_l} \leftarrow \mathcal{D}$;
            end
            $j += 1$;
        end
        if $\mathcal{D}_{E_u E_l} \leq \mathcal{D}_{iE_l}$ then
            $\mathcal{D}_{E_u E_l} \leftarrow \mathcal{D}_{iE_l}$;
            $\vec{V_{ul}} = \vec{\mathcal{P}_{u_i}\mathcal{P}_{l_j}}$;
            $ind_u \leftarrow i$;
            $ind_l \leftarrow j$;
        end
        $i += 1$;
    end
    return $\vec{V_{ul}}, ind_u, ind_l$;
end
```

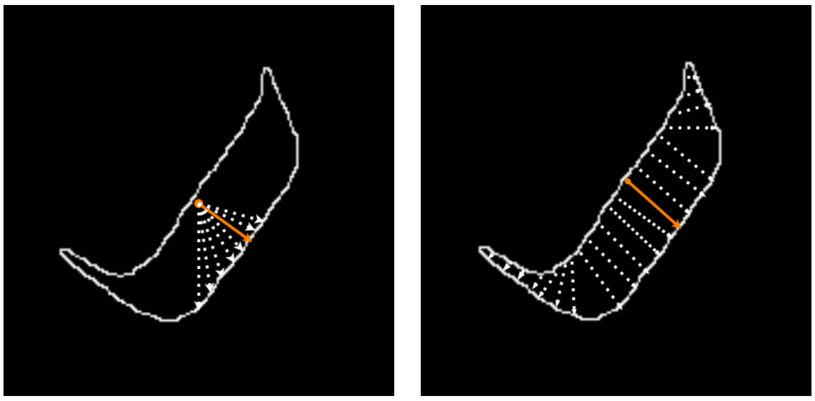
This calibration technique facilitates accurate object positioning by accounting for the spatial relationships between the upper and lower components. However, the efficacy of Algorithm 1 can be compromised by suboptimal segmentations. To mitigate this issue, we introduce Algorithm 2 as a solution. In essence, if the vector V_{ul} derived from Algorithm 1 is accurate, then upon projecting the upper line E_u along the translation vector, the distance between the resultant line $\hat{S_u}$ and the lower line S_l should be approximately, or equal, to 0.

Leveraging this concept, we construct the set of neighboring vectors $\mathbb{V}$ by maintaining the initial point of the vector V_{ul} unchanged and selecting k neighboring points for the terminal point. Subsequently, we project the upper line following each translation vector in $\mathbb{V}$ and choose the vector that results in the smallest distance.



(a) Input image **(b)** Segmentation **(c)** Bilateral filtering **(d)** Edge Extraction

Figure 5. The progress of object segmentation and edge extraction



(a) Point to line translation vector **(b)** Line to line translation vector

Figure 6. Illustration of the estimated translation vector

This novel algorithm serves to enhance the robustness of the calibration process against suboptimal segmentations, ensuring more reliable results.

4. Experimental Results

4.1. Experiment Setup

To evaluate the proposed method for estimating the location of objects, we conducted experiments using a vision system with a camera positioned above, parallel to the floor, and at a distance of $40cm$. The real-world coordinates were defined using a checkerboard pattern, where each square was measured $3 \times 3cm^2$. Further details about the experimental setup, including hardware specifications, are provided in Table 1. In our experiments, we used a specific type of object to evaluate the performance of our localization algorithm. The object selected for this study is a standard electrical charger, commonly found in households and industrial settings. This object was chosen due to its well-defined shape and easily recognizable features, which facilitate accurate detection and segmentation. The dimensions of this object are $4.5cm \times 3.0cm \times 3.5cm$. The dimensions, shape, and sharpness of the localized object significantly impact the performance of the localization algorithm. The object's well-defined edges and distinct features enable the deep learning models to accurately detect and segment it from the background. The size of the object ensures that it is neither too small to be overlooked by the detection model.

The rectangular shape with flat surfaces and sharp edges facilitates precise boundary detection during the segmentation process. The clear and distinct contours of the object enhance the segmentation accuracy, leading to better calibration results.

The evaluation metrics employed for a comprehensive assessment are Intersection over Union (IOU) for object segmentation, Mean Average Precision (mAP) for object detection, and the Euclidean metric for geometric accuracy.

Metrics: We utilize a set of robust metrics to assess the performance of our proposed object localization method thoroughly. The Intersection over Union (IOU) metric, pivotal in the object segmentation phase, is defined as the ratio of the area of overlap (A_{ol}) between predicted (A_{pred}) and ground truth (A_{gt}) bounding boxes to the area of union (A_{un}):

$$IOU = \frac{A_{ol}}{A_{un}} = \frac{A_{ol}}{A_{pred} + A_{gt} - A_{ol}} \tag{5}$$

The mean average precision (mAP) is utilized for the object detection phase, calculated by integrating the precision (P) over the recall (R) and the number of classes N at various IOU thresholds:

$$mAP = \frac{1}{\mathrm{N}} \sum_{i=0}^{N} AP = \frac{1}{\mathrm{N}} \sum_{i=0}^{N} \int_{0}^{1} P(R), d_R \tag{6}$$

Algorithm 2: Translation Vector Correction

```
Input: $\mathcal{P}_u = E_{u_1}, E_{u_2}, \dots, E_{u_N}$
       $\mathcal{P}_l = E_{l_1}, E_{l_2}, \dots, E_{l_M}$
       $N$ is the number of points in line $E_u$
       $M$ is the number of points in line $E_l$
       $\vec{V_{ul}}, ind_u, ind_l \leftarrow Algorithm1$
Output: Translation vector: $\vec{V_{ul}}$
begin:
        ▷ Choose $k$ neighboring points of $\mathcal{P}_{l_{ind_l}}$
    $i \leftarrow -k/2$ ;
    $\mathbb{V} \leftarrow \{\}$;          ▷ Set of translation vectors
    while $i < k/2$ do
        $j \leftarrow ind_l + i$;
        $\mathbb{V}.pushback(\overrightarrow{\mathcal{P}_{u_{ind_u}}\mathcal{P}_{l_j}})$;   ▷ Pushback
          the neighboring vectors of $\vec{V_{ul}}$ into $\mathbb{V}$
    end
    $\mathcal{D}_{min} \leftarrow +\infty$;
    for $j = 0; j < size(\mathbb{V}); j++$ do
        $\bar{\mathcal{P}_u} \leftarrow \mathcal{P}_u + \mathbb{V}_i$;     ▷ Project upper line
          following the translation vector
          ▷ Calculate the distance between $\bar{\mathcal{P}_u}$
          and $\mathcal{P}_l$
        $\mathcal{D} \leftarrow 0$;
        $n \leftarrow 0$ ;
        $m \leftarrow 0$ ;
        while $n < (N-1)$ do
            while $m < (M-1)$ do
                Calculate the distance $d$
                  between two points
                  $(\bar{\mathcal{P}}_{u_n} = (\bar{x}_{u_n}, \bar{y}_{u_n}))$ and
                  $(\mathcal{P}_{l_m} = (x_{l_m}, y_{l_m}))$ based on
                  (4);
                $D \leftarrow D + d$;
                $m+ = 1$;
            end
            $n+ = 1$;
        end
        if $\mathcal{D}_{min} \geq \mathcal{D}$ then
            $\mathcal{D}_{min} \leftarrow \mathcal{D}$;
            $\vec{V_{ul}} \leftarrow \mathbb{V}_j$;
        end
    end
    return $\vec{V_{ul}}$;
end
```

Table 1. Experiment setup details

Parameter	Spec
Process	Intel Xeon Processor with two cores @ 2.3 GHz
GPU	NVIDIA Tesla T4
RAM	13 GB
OS	Ubuntu 20.04 LTS

Table 2. Performance comparison of various object detection models

Algorithm	mAP	Pr	Rc	MS
RTMDet [37]	96.9%	94.5%	93.1%	52.3
MobileNet [35]	94.8%	93.8%	93.4%	**4.6**
Fast R-CNN [34]	97.0%	93.4%	94.1%	12.9
Yolov3 [33]	96.3%	95.8%	95.7%	8.7
Yolov4 [36]	96.8%	96.6%	95.4%	60.0
Yolov7 [38]	97.1%	95.7%	93.1%	37.2
Yolov8 [39]	97.8%	95.5%	94.4%	11.1
Our	**98.7%**	**98.6%**	**97.0%**	7.0

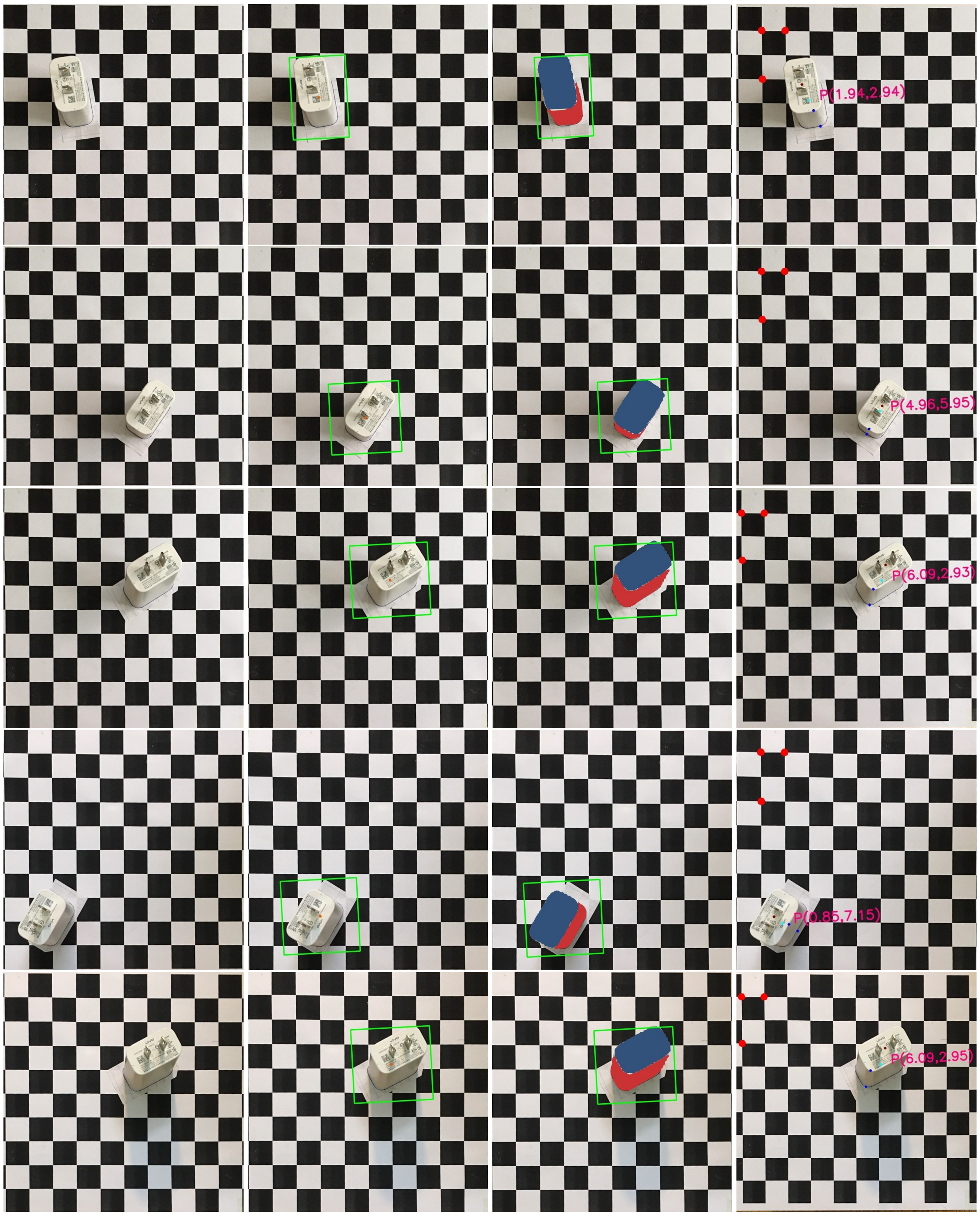
The Euclidean metric assesses geometric accuracy by calculating the Euclidean distance $d_{Euclidean}$ between the predicted $((x_{pred}, y_{pred}))$ and true $((x_{tr}, y_{tr}))$ object coordinates:

$$d_{Euclidean} = \sqrt{(x_{pred} - x_{tr})^2 + (y_{pred} - y_{tr})^2} \quad (7)$$

Datasets: We acquired three distinct datasets, each corresponding to a specific phase of our paper. In the object detection phase, we amassed a collection of 400 images captured from various object locations. To ensure a comprehensive evaluation, we partitioned this dataset into three subsets: a training set comprising 60% of the data randomly selected, a validation set with 30%, and a test set with the remaining 10%. Subsequently, in the object detection phase, we gathered 5000 images from diverse object locations. To maintain a robust evaluation approach, we split this dataset into a training set (70% of the data randomly selected), a validation set (20%), and a test set (10%). Lastly, for the object localization phases, we gathered 60 images, distributed into 10 folds, each containing 6 images.

4.2. Object Detection and Object Segmentation Results

Table 2 presents the results of various detection models evaluated in terms of mean average precision – mAP, model size – MS (MB), precision – Pr, and recall – Rc. The Yolov3 [33] model achieves a mAP of 90.0% with a model size of 8.7 MB, accompanied by precision and recall scores of 85.9% and 84.6%, respectively. In contrast, the Fast R-CNN [34] model demonstrates superior performance with a mAP of 97.0% despite a larger model size of 12.9 MB, achieving precision and recall scores of 93.4% and 42.1%, respectively. The MobileNet [35] model offers a mAP of 94.8% with a relatively compact model size of 4.6 MB, achieving precision and recall scores of 93.8% and 93.4%, respectively. The Yolov4 [36] and RTMDet [37] models exhibit competitive mAP scores of 96.8% and 96.9%, respectively, with larger model sizes of 60.0 MB and 52.3 MB. The Yolov7 [38] and Yolov8 [39] models can achieve superior accuracy with a mAP of 97.1% and 97.8%, respectively, with a precision of 95.7% and 95.5%, and a recall of 93.1% and 94.4%, respectively. Our proposed model outperforms the others with an mAP of 98.7%, precision of 98.6%, and recall of 97.0%.



(a) Input image **(b)** Object detection **(c)** Segmentation **(d)** Calibiration result

Figure 7. Visualized examples of experimental results: figure (b): the orange point is the Yolo center, figure (d): dark red is the upper part center. The vector created by the blue points is a translation vector; the light blue point is the correction center

Additionally, our model exhibits a relatively compact size of 7.0 MB compared to other models, indicating its efficiency in terms of memory usage. These results underscore the effectiveness and efficiency of our proposed object detection model for accurately detecting objects in various scenarios, making it well-suited for practical deployment in real-world applications.

Table 3. Experimental results evaluate the position error of our algorithm (mm)

Fold	Sample	Traditional Method			Regression Method [30]			Proposed Method					
								Before Correction			After Correction		
		Δx	Δy	Err	Δx	Δy	Err	Δx	Δy	Err	Δx	Δy	Err
	1	9.69	5.51	11.15	1.34	1.32	1.88	**0.38**	1.14	**1.20**	0.95	**0.76**	1.22
	2	8.36	8.74	12.09	1.92	2.50	3.15	3.04	**1.33**	3.32	**1.71**	**1.33**	**2.17**
1	3	5.13	8.93	10.30	1.23	1.52	1.96	**1.14**	**0.95**	**1.48**	**1.14**	**0.95**	**1.48**
	4	10.07	8.36	13.09	1.79	1.97	2.66	**1.14**	**1.33**	**1.75**	1.52	**1.33**	2.02
	5	8.55	10.26	13.36	0.75	1.12	1.35	0.76	**0.19**	0.78	**0.38**	0.57	**0.69**
	6	9.31	9.31	13.17	0.96	1.41	1.71	**0.19**	1.52	1.53	0.57	**0.76**	**0.95**
	1	4.18	5.89	7.22	1.45	3.07	3.40	1.33	2.09	2.48	**0.95**	**1.14**	**1.48**
	2	10.07	13.11	16.53	3.12	3.41	4.62	**2.47**	1.71	3.00	**2.47**	**1.33**	**2.81**
2	3	3.42	7.03	7.82	1.21	2.78	3.03	1.52	1.90	2.43	**0.76**	**0.57**	**0.95**
	4	10.07	8.74	13.33	2.13	1.51	2.61	**0.95**	**0.19**	**0.97**	1.90	0.95	2.12
	5	11.02	12.16	16.41	2.94	2.67	3.97	**2.28**	1.71	2.85	2.47	**0.57**	**2.53**
	6	8.93	11.02	14.18	0.43	1.34	1.41	1.33	0.38	1.38	**0.19**	**0.19**	**0.27**
	1	9.12	7.60	11.87	1.39	1.42	1.99	**0.95**	**0.19**	**0.97**	1.14	0.38	1.20
	2	4.18	13.87	14.49	2.54	3.36	4.21	3.23	**0.19**	3.24	**2.28**	0.57	**2.35**
3	3	9.88	4.18	10.73	0.76	1.12	1.35	**0.57**	**0.76**	**0.95**	**0.57**	**0.76**	**0.95**
	4	7.60	8.93	11.73	1.89	2.17	2.88	**1.71**	**0.95**	**1.96**	**1.71**	**0.95**	**1.96**
	5	10.45	4.94	11.56	0.88	2.34	2.50	1.71	1.52	2.29	**0.57**	**0.76**	**0.95**
	6	6.08	7.60	9.73	0.36	0.57	0.67	**0.19**	**0.57**	**0.60**	**0.19**	**0.57**	**0.60**
	1	13.87	10.26	17.25	1.37	2.84	3.15	**0.76**	2.47	2.58	0.95	**2.09**	**2.30**
	2	11.97	7.98	14.39	2.84	1.45	3.19	**2.28**	**1.33**	**2.64**	**2.28**	**1.33**	**2.64**
4	3	5.32	4.56	7.01	0.35	0.81	0.88	**0.19**	**0.38**	**0.42**	**0.19**	0.57	0.60
	4	5.51	16.34	17.24	0.32	1.32	**1.36**	0.76	**1.14**	1.37	**0.19**	1.71	1.72
	5	10.26	8.36	13.23	0.92	0.92	1.30	2.28	0.76	2.40	**0.57**	**1.14**	**1.27**
	6	6.27	11.40	13.01	1.47	1.63	2.19	1.90	2.85	3.43	**1.33**	**1.33**	**1.88**
Average		8.30	8.96	12.54	1.43	1.86	2.34	1.38	1.15	1.92	**1.12**	**0.94**	**1.55**

Table 4. Performance comparison of various object segmentation models

Algorithm	**mAP**	**Pr**	**Rc**	**MS** (MB)
Yolov5 [40]	98.7%	97.1%	96.2%	**7.4**
RCNN [41]	97.8%	98.1%	96.4%	16.8
Yolov7 [38]	99.0%	99.0%	97.8%	37.9
Yolov8 [39]	99.2%	98.7%	97.4%	11.8
Our	**99.8%**	**99.1%**	**97.9%**	28.9

A comparative analysis of various object segmentation models is presented in 4. Among the models evaluated, Yolov5 [40] emerges as a strong contender, showcasing a notable mAP@0.5 score of 98.7%, a precision rate of 97.1%, and a recall rate of 96.2%. These metrics indicate its robust ability to accurately identify objects within images while maintaining a relatively compact model size of 7.4 MB, making it an efficient choice for resource-constrained environments. In addition, Yolov7 [38] and Yolov8 [39] demonstrate a commendable performance with high mAP scores of 99.0% and 99.2%, respectively, along with impressive precision and recall values. However, what sets our proposed model apart is its exceptional performance across all metrics. With an outstanding mAP@0.5 score of 99.8%, precision rate of 99.1%, and recall rate of 97.9%, our model surpasses all others in terms of segmentation accuracy while maintaining a moderate model size of 28.9 MB. These results underscore the efficacy of our segmentation model in accurately delineating objects within images.

Figure 7 visually illustrates the experimental results of our proposed method, including object detection, object segmentation, and object calibration. These visualizations provide a comprehensive insight into the efficacy and accuracy of each phase of our methodology. Object detection showcases the ability of our model to accurately identify and localize objects within the scene, laying the foundation for subsequent processing steps. Object segmentation highlights the precision with which our algorithm delineates the boundaries of detected objects, ensuring accurate localization and analysis. Finally, object calibration visually demonstrates the refinement and optimization of object positions based on real-world coordinates, validating the effectiveness of our calibration approach in enhancing spatial accuracy.

4.3. Object Localization Results

The experimental results in Table 3 offer quantitative insights into the performance comparison between our proposed method, the traditional approach, and the Regression-based method [30], our previous method [30] (Regression-based method). Across multiple folds and samples, our method consistently demonstrates superior performance in terms of position error metrics.

Table 5. Processing time of our proposed method (milliseconds)

Phase	**Processing Time**
Object Detection	15 ± 2
Object Segmentation	40 ± 5
Object Calibration	300 ± 10

For instance, in Fold 1, Sample 1, the traditional method yields position errors of $\Delta x = 9.69$ mm and $\Delta y = 5.51$ mm, while our proposed method achieves significantly lower errors of $\Delta x = 0.38$ mm and $\Delta y = 1.14$ mm before correction, and $\Delta x = 0.95$ mm and $\Delta y = 0.76$ mm after correction.

The average position errors across all folds and samples further highlight the effectiveness of our proposed method. On average, our method achieves position errors of $\Delta x = 1.12$ mm and $\Delta y = 0.94$ mm after correction, compared to $\Delta x = 8.30$ mm and $\Delta y = 8.96$ mm for the traditional method, which reduces the position error by 87.64%. Similarly, the regression-based method yields average errors of $\Delta x = 1.43$ mm and $\Delta y = 1.86$ mm, indicating a noticeable improvement over the traditional approach but still inferior to our proposed method.

These quantitative results underscore the significant reduction in position errors achieved by our proposed method compared to both traditional and regression-based approaches. The superior accuracy and precision offered by our method are particularly advantageous in applications where precise object localization is paramount, such as robotic manipulation, augmented reality, and autonomous navigation systems.

The processing time for each phase of our proposed method is summarized in Table 5. In the object detection phase, our algorithm takes approximately 15 ± 2 milliseconds to detect objects within the camera's field of view. Subsequently, during the object segmentation phase, which involves segmenting the detected objects into upper and lower planes, the algorithm also requires around 40 ± 5 milliseconds. Finally, in the object calibration phase, where the precise position of the objects is determined based on the segmented data, the processing time remains consistent at approximately 300 ± 10 milliseconds. This efficient processing time across all phases underscores the real-time applicability and practical feasibility of our proposed method for object localization in industrial vision systems.

5. Conclusion

In conclusion, we have presented M-Calib, a comprehensive methodology for precise object localization and calibration leveraging advanced computer vision techniques for industrial robot vision systems.

Through the integration of advanced computer vision techniques, including object detection, segmentation, and calibration, our proposed approach offers a robust and accurate solution for determining the real-world positions of objects. Experimental results demonstrate the effectiveness of our method in significantly reducing 87.65% position errors compared to traditional approaches, thereby enhancing spatial accuracy in industrial environments. Furthermore, the computational efficiency of our method, as evidenced by minimal processing times, underscores its practical viability for real-world deployment. Overall, our proposed methodology holds promise for a wide range of industrial applications where precise object localization is essential, offering a reliable solution to optimize operational efficiency and enhancing productivity. In the future, our aim is to explore machine learning techniques for automatic calibration parameter adjustment and extend our methodology to support real-time dynamic object localization.

AUTHORS
Thanh Nguyen Canh – Department of Robotics Engineering, University of Engineering and Technology, Vietnam National University, Xuan Thuy, Hanoi, 10000, Vietnam, e-mail: canhthanh@vnu.edu.vn.
Du Trinh Ngoc – Department of Robotics Engineering, University of Engineering and Technology, Vietnam National University, Xuan Thuy, Hanoi, 10000, Vietnam, e-mail: ngocdu2105@gmail.com.
Xiem HoangVan* – Department of Robotics Engineering, University of Engineering and Technology, Vietnam National University, Xuan Thuy, Hanoi, 10000, Vietnam, e-mail: xiemhoang@vnu.edu.vn.

*Corresponding author

ACKNOWLEDGEMENTS

Thanh Nguyen Canh was funded by the Master, PhD Scholarship Programmer of Vingroup Innovation Foundation (VINIF), code VINIF.2023.ThS.120

References

[1] M. Salah, A. Ayyad, M. Humais, D. Gehrig, A. Abusafieh, L. Seneviratne, D. Scaramuzza, and Y. Zweiri, "E-calib: A fast, robust and accurate calibration toolbox for event cameras", *IEEE Transactions on Image Processing*, vol. 33, 2024, 3977–3990, doi: 10.1109/TIP.2024.3410673.

[2] Y. Cai, F. Wang, X. Wang, S. Li, Y. Wang, J. Yang, T. Yan, X. Zhan, F. Wang, R. Cheng, et al., "Broadband visual adaption and image recognition in a monolithic neuromorphic machine vision system", *Advanced Functional Materials*, vol. 33, no. 5, 2023, 2212917, doi: 10.1002/adfm.202212917.

[3] M. Salah, M. Chehadah, M. Humais, M. Wahbah, A. Ayyad, R. Azzam, L. Seneviratne, and Y. Zweiri, "A neuromorphic vision-based measurement for robust relative localization in future space exploration missions", *IEEE Transactions on Instrumentation and Measurement*, vol. 73, 2022, 1–12, doi: 10.1109/TIM.2022.3217513.

[4] L.-E. Montoya-Cavero, R. D. de León Torres, A. Gómez-Espinosa, and J. A. E. Cabello, "Vision systems for harvesting robots: Produce detection and localization", *Computers and electronics in agriculture*, vol. 192, 2022, 106562, doi: 10.1016/j.compag.2021.106562.

[5] N. Robinson, B. Tidd, D. Campbell, D. Kulić, and P. Corke, "Robotic vision for human-robot interaction and collaboration: A survey and systematic review", *ACM Transactions on Human-Robot Interaction*, vol. 12, no. 1, 2023, 1–66, doi: 10.1016/j.compag.2021.106562.

[6] P. Rosenberger, A. Cosgun, R. Newbury, J. Kwan, V. Ortenzi, P. Corke, and M. Grafinger, "Object-independent human-to-robot handovers using real time robotic vision", *IEEE Robotics and Automation Letters*, vol. 6, no. 1, 2020, 17–23, doi: 10.1109/LRA.2020.3026970.

[7] G. D'Emilia and D. Di Gasbarro, "Review of techniques for 2d camera calibration suitable for industrial vision systems". In: *Journal of Physics: Conference Series*, 2017, 012030, doi: 10.1088/1742-6596/841/1/012030.

[8] Y.-J. Zhang. "Camera calibration". In: *3-D Computer Vision: Principles, Algorithms and Applications*, 37–65. Springer, 2023.

[9] C. Ricolfe-Viala and A. Esparza, "The influence of autofocus lenses in the camera calibration process", *IEEE Transactions on Instrumentation and Measurement*, vol. 70, 2021, 1–15, doi: 10.1109/TIM.2021.3055793.

[10] J.-H. Chuang, C.-H. Ho, A. Umam, H.-Y. Chen, J.-N. Hwang, and T.-A. Chen, "Geometry-based camera calibration using closed-form solution of principal line", *IEEE Transactions on Image Processing*, vol. 30, 2021, 2599–2610, doi: 10.1109/TIP.2020.3048684.

[11] S.-E. Lee, K. Shibata, S. Nonaka, S. Nobuhara, and K. Nishino, "Extrinsic camera calibration from a moving person", *IEEE Robotics and Automation Letters*, vol. 7, no. 4, 2022, 10344–10351, doi: 10.1109/LRA.2022.3192629.

[12] B. Chen, Y. Liu, and C. Xiong, "Automatic checkerboard detection for robust camera calibration". In: *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021, 1–6, doi: 10.1109/ICME51207.2021.9428389.

[13] S. Liang and Y. Zhao, "Common pole-polar and common tangent properties of dual coplanar circles and their application in camera calibration", *Multimedia Tools and Applications*, vol. 83, no. 1, 2024, 381–401, doi: 10.1007/s11042-023-15684-4.

[14] S. Liang and Y. Zhao, "Camera calibration based on the common pole-polar properties between two coplanar circles with various positions", *Applied Optics*, vol. 59, no. 17, 2020, 5167–5178, doi: 10.1364/AO.388109.

[15] Y. Xie, R. Shao, P. Guli, B. Li, and L. Wang, "Infrastructure based calibration of a multi-camera and multi-lidar system using apriltags". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, 605–610, doi: 10.1109/IVS.2018.8500646.

[16] A. Wu, H. Xiao, and F. Zeng, "A camera calibration method based on opencv". In: *Proceedings of the 4th International Conference on Intelligent Information Processing*, 2019, 320–324, doi: 10.1145/3378065.3378127.

[17] A. Fetić, D. Jurić, and D. Osmanković, "The procedure of a camera calibration using camera calibration toolbox for matlab". In: *2012 Proceedings of the 35th International Convention MIPRO*, 2012, 1752–1757.

[18] C. B. Duane, "Close-range camera calibration", *Photogramm. Eng*, vol. 37, no. 8, 1971, 855–866.

[19] J. Fryer, T. Clarke, and J. Chen, "Lens distortion for simple c-mount lenses", *International Archives of Photogrammetry and remote sensing*, vol. 30, 1994, 97–101.

[20] M.-T. Lu and J.-H. Chuang, "Fully automatic camera calibration for principal point using flat monitors". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, 3154–3158, doi: 10.1109/ICIP.2018.8451222.

[21] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses", *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, 1987, 323–344, doi: 10.1109/JRA.1987.1087109.

[22] M. Gašparović and D. Gajski, "Two-step camera calibration method developed for micro uav's", *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 41, 2016, 829–833, doi: 10.5194/isprs-archives-XLI-B1-829-2016.

[23] X. Chen, R. Fan, J. Wu, X. Song, Q. Liu, Y. Wang, Y. Wang, and B. Tao, "Fourier-transform-based two-stage camera calibration method with simple periodical pattern", *Optics and Lasers in Engineering*, vol. 133, 2020, 106121, doi: 10.1016/j.optlaseng.2020.106121.

[24] Y. I. Abdel-Aziz, H. M. Karara, and M. Hauck, "Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry", *Photogrammetric engineering & remote sensing*, vol. 81, no. 2, 2015, 103–107, doi: 10.14358/PERS.81.2.103.

[25] Z. Zhao, D. Ye, X. Zhang, G. Chen, and B. Zhang, "Improved direct linear transformation for parameter decoupling in camera calibration", *Algorithms*, vol. 9, no. 2, 2016, 31, doi: 10.3390/a9020031.

[26] F. Barone, M. Marrazzo, and C. J. Oton, "Camera calibration with weighted direct linear transformation and anisotropic uncertainties of image control points", *Sensors*, vol. 20, no. 4, 2020, 1175, doi: 10.3390/s20041175.

[27] Z. Shi, Y. Shang, X. Zhang, and G. Wang, "Dlt-lines based camera calibration with lens radial and tangential distortion", *Experimental Mechanics*, vol. 61, no. 8, 2021, 1237–1247, doi: 10.1007/s11340-021-00726-5.

[28] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations". In: *Proceedings of the seventh ieee international conference on computer vision*, 1999, 666–673, doi: 10.1109/ICCV.1999.791289.

[29] T. H. M. Siddique, Y. Rehman, T. Rafiq, M. Z. Nisar, M. S. Ibrahim, and M. Usman, "3d object localization using 2d estimates for computer vision applications". In: *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*, 2021, 1–6, doi: 10.1109/MAJICC53071.2021.9526270.

[30] X. H. Van and N. Do, "An efficient regression method for 3d object localization in machine vision systems", *IAES International Journal of Robotics and Automation*, vol. 11, no. 2, 2022, 111, doi: 10.11591/ijra.v11i2.pp111-121.

[31] R. L. Van Hook. *A Comparison of Monocular Camera Calibration Techniques*. PhD thesis, Wright State University, 2014.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, 779–788.

[33] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement", *arXiv preprint arXiv:1804.02767*, 2018.

[34] R. Girshick, "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, 2015, 1440–1448.

[35] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications", *arXiv preprint arXiv:1704.04861*, 2017.

[36] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection", *arXiv preprint arXiv:2004.10934*, 2020.

[37] C. Lyu, W. Zhang, H. Huang, Y. Zhou, Y. Wang, Y. Liu, S. Zhang, and K. Chen, "Rtmdet: An empirical study of designing real-time object detectors", *arXiv preprint arXiv:2212.07784*, 2022.

[38] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new

state-of-the-art for real-time object detectors". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, 7464–7475.

[39] G. Jocher, A. Chaurasia, and J. Qiu. "Ultralytics yolov8", January 2023.

[40] G. Jocher. "Ultralytics yolov5", 2020.

[41] 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, 580–587.