robotics

MDPI

Article

Online Self-Supervised Learning for Accurate Pick Assembly Operation Optimization

Sergio Valdés [1,*], Marco Ojer [1,2] and Xiao Lin [1]

[1] Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), 20009 Donostia-San Sebastian, Spain; mojer@vicomtech.org (M.O.); xlin@vicomtech.org (X.L.)
[2] Computer Science and Artifitial Intelligence Department, University of the Basque Country (UPV/EHU), 20018 Donostia-San Sebastian, Spain
* Correspondence: svaldes@vicomtech.org

Abstract: The demand for flexible automation in manufacturing has increased, incorporating vision-guided systems for object grasping. However, a key challenge is in-hand error, where discrepancies between the actual and estimated positions of an object in the robot's gripper impact not only the grasp but also subsequent assembly stages. Corrective strategies used to compensate for misalignment can increase cycle times or rely on pre-labeled datasets, offline training, and validation processes, delaying deployment and limiting adaptability in dynamic industrial environments. Our main contribution is an online self-supervised learning method that automates data collection, training, and evaluation in real time, eliminating the need for offline processes. Building on this, our system collects real-time data during each assembly cycle, using corrective strategies to adjust the data and autonomously labeling them via a self-supervised approach. It then builds and evaluates multiple regression models through an auto machine learning implementation. The system selects the best-performing model to correct the misalignment and dynamically chooses between corrective strategies and the learned model, optimizing the cycle times and improving the performance during the cycle, without halting the production process. Our experiments show a significant reduction in the cycle time while maintaining the performance.

Keywords: self-supervised learning; online learning; vision-guided systems; pick-and-place; in-hand error; peg-in-hole; cycle time optimization

Academic Editor: Marco Ceccarelli

Received: 20 November 2024
Revised: 20 December 2024
Accepted: 27 December 2024
Published: 30 December 2024

Citation: Valdés, S.; Ojer, M.; Lin, X. Online Self-Supervised Learning for Accurate Pick Assembly Operation Optimization. *Robotics* **2025**, *14*, 4. https://doi.org/10.3390/robotics14010004

1. Introduction

In recent years, the demand for automation in manufacturing and assembly processes has surged, driven by the need for increased efficiency, accuracy, and flexibility. Vision-guided pick-and-assembly systems allow robots to identify, grasp, and assemble components with minimal human intervention. However, despite advancements in robotics and computer vision, challenges remain, particularly concerning the grasping accuracy [1].

Grasping inaccuracies, defined as a discrepancy between the expected and actual positions of an object within a robot's gripper, are often referred to in the literature as in-hand errors. This error may arise from various sources, including the vision system, camera calibration, or slippage during grasping. These inaccuracies not only impact the immediate success of the grasp but can also hinder subsequent assembly operation. To achieve accurate assemblies in the presence of in-hand errors, corrective strategies are normally involved. These strategies can be categorized into two types: picking refinement strategies and flexible assembly strategies.

Picking refinement strategies focus on enhancing the picking process to ensure grasping while mitigating primary sources of error. One such approach involves segmenting the picking process into multiple stages [2,3]. However, for accuracy-demanding tasks such as peg-in-hole assemblies, relying solely on picking strategies may be insufficient, as residual in-hand errors may still occur. Thus, the implementation of flexible assembly strategies capable of absorbing this error is necessary. A possible strategy involves moving the robot along a spiral path after the peg makes contact with the hole surface, guiding it to successful insertion [4,5].

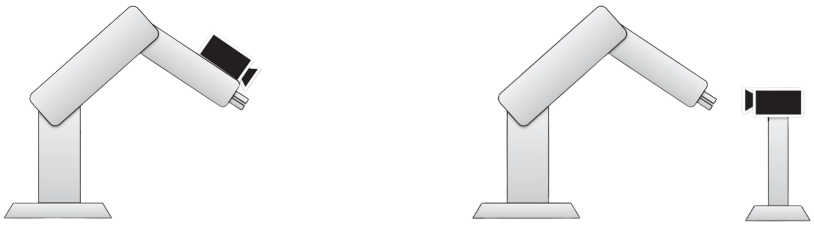
Some corrective strategies involve additional steps that lengthen the production cycles, while others employ machine or deep learning to optimize the process. However, the latter often require significant time and resources for dataset labeling, offline training, and validation, which delays deployment and limits adaptability in dynamic industrial environments. To overcome these challenges, our main contribution is a pipeline that automates data collection, training, and evaluation processes without human supervision, leveraging online implementation, self-supervised data labeling, and AutoML. Designed for pick-and-place assembly tasks with an eye-in-hand configuration, this method aims to reduce cycle times and streamline the training process.

This paper is structured as follows: Section 2 reviews relevant research, Section 3 describes our proposed method, Section 4 presents the experimental findings, and Section 5 discusses the broader implications.

2. State of the Art

The vision-guided pick-and-assembly operation consist of identifying and grasping a component and accurately inserting it into a hole. The assembly stage is known as peg-in-a-hole problem, and this topic has been studied in the literature [4–13]. This process relies on the accuracy of the grasp during the pick stage. Any error in the grasp compromises the entire operation, which becomes even more critical in high-accuracy tasks such as assembling tightly fitting components, where smaller tolerances further increase the complexity.

Focusing on the initial stage of detection and grasping, cameras are the most common sensors used to provide information about the location and orientation of objects. A camera can either be fixed in a specific position (eye-to-hand configuration) or mounted on the robot's end-effector (eye-in-hand configuration), capturing images from the tool's perspective. These configurations are shown in Figure 1.



(**a**) (**b**)

Figure 1. Representation of common camera configurations: (**a**) eye-in-hand configuration; (**b**) fixed eye–hand configuration [14].

Regardless of the camera's configuration, the location of the target object is initially obtained within the camera's coordinate system. It is necessary to represent the information within the robot's coordinate system. This process, known as hand–eye calibration, requires the estimation of the relationship between the two coordinate systems. The accurate estimation of such a transformation matrix for robotic arms is a challenging task [15,16]. Hand–eye calibration methods are classified into separate and simultaneous approaches. Separate methods, such as those presented by Tsai et al. [17], Chou et al. [18], and

Park et al. [19], estimate the rotation and translation parameters independently, which can lead to error propagation. In contrast, simultaneous methods, such as those described by Daniilidis et al. [20], Lu et al. [21], and Li et al. [22], estimate both parameters at the same time, making them more robust to rotation noise. The study by Enebuse et al. [15] evaluates the effectiveness of these six hand–eye calibration algorithms, showing that the accuracy of each method varies based on the number of robot movements during calibration and the level of noise present. Additionally, the performance of separate methods tends to be affected by rotation noise, while simultaneous methods exhibit noise transfer between rotation and translation. This analysis underscores the importance of selecting the appropriate method based on the emphasis placed on either orientation or translation accuracy.

Camera calibration establishes the foundation for accuracy in vision-guided pick systems. However, its effectiveness is also contingent upon the grasping methods employed. For unknown objects, a highly researched area in robotics, the most common techniques in the literature involve using object recognition and the extraction of specific features to estimate grasp poses [23,24]. In the case of known objects, the grasping task simplifies to estimating the object's pose, identifying a grasp point, and determining the pose transformation during the grasping process. This task can also be improved through other strategies. One such approach involves segmenting the grasping process into multiple stages [2,3]. All strategies that ensure successful grasping while mitigating primary sources of error are referred to as **picking refinement strategies.** Nevertheless, inherent errors in these methods, along with calibration inaccuracies, can lead to discrepancies between the expected and actual positions of the object in the robot's gripper, which directly affects the success of high-accuracy assembly tasks.

Traditionally, assembly systems in the industry relied on classical offline programming where robot poses are limited to a set of predefined poses, which are manually taught by an skilled operator. This approach is effective when the component is always picked in the same manner. However, it becomes inadequate in flexible production lines, where the grasping pose is not constant and must be calculated at each cycle, complicating the assembly process.

To address these challenges, several **flexible assembly strategies** have been proposed, including methods such as spiral search or Lissajous curves, which employ simple heuristics and force feedback to guide the insertion process. For instance, Park et al. [4,5] use a spiral trajectory to search for the hole, while Jokesch et al. [6] apply a Lissajous curve. These approaches rely on predefined paths and iterative adjustments to the insertion force, enabling the robot to compensate for misalignments during assembly. However, they introduce additional steps that increase the cycle times.

To overcome the limitations of these heuristic methods, recent research has focused on deep learning (DL) and reinforcement learning (RL). These methods often combine visual information with force feedback to train models capable of reducing the cycle times while improving the accuracy and flexibility. For example, Triyonoputro et al. [8] employ a multi-view deep learning approach to guide peg insertion into the hole, demonstrating the potential of vision-based strategies. Similarly, Abdullah et al. [9] integrate DL with hybrid vision/force control to achieve precise pick-and-assembly operations, while Almaghout et al. [11] utilize a vision-based DL framework to minimize misalignment errors during assembly. On the other hand, reinforcement learning (RL) has shown significant potential in tackling complex sequential tasks. For instance, Schoettler et al. [12] successfully apply RL to tight-clearance peg-in-hole tasks, training recurrent neural networks to adapt the insertion trajectories dynamically. Yasutomi et al. [13] further demonstrate the

effectiveness of RL in learning fine-grained control strategies for accuracy tasks, even under varying environmental conditions.

Despite their promise, these methods require significant resources, including large labeled datasets and extensive training and validation times. This dependency on offline model preparation complicates their application in dynamic and fast-paced industrial settings.

To address these limitations, self-supervised learning (SSL) has emerged as a helpful approach. SSL enables systems to autonomously label data from raw observations, reducing the need for pre-labeled datasets. For example, Haugaard et al. [25] focus on combining spiral search with visual feedback to collect and label data using an SSL paradigm, training a neural network offline and deploying it based on the validation results. While this reduces the cycle times for assembly tasks, their offline implementation requires delaying the production process to develop and validate the model.

A related application of SSL in the context of grasping is explored by Peng et al. [26], who propose a self-supervised learning-based 6-DOF grasp planning method. In their approach, grasp poses are automatically labeled using a force closure decision algorithm, and a deep learning model is trained offline to classify the grasp quality. While this method enhances the robustness in grasping unknown objects in unstructured environments, its offline nature limits its adaptability to dynamic settings.

These examples highlight two corrective strategies: a flexible assembly strategy [25], which focuses on compensating for in-hand errors during assembly, and a picking refinement strategy [26], aimed at improving the grasping accuracy during the picking stage. While effective, both rely on offline implementations that require delaying the production process to develop and validate models, resulting in downtime and limited adaptability. To overcome these challenges, an alternative is to implement advanced learning techniques in an online manner, enabling continuous training and updates while the production line is operational. This approach enhances the performance and reduces the cycle times by supporting real-time decision-making, processing data as they become available. A challenge in online learning is the unpredictability and dynamism of data distributions, complicating model selection and hyperparameter optimization. Automated machine learning (AutoML) addresses this by automating processes like model selection and tuning, allowing systems to adapt to fluctuating data with minimal intervention. AutoML can train machine learning models by exploring various configurations and hyperparameters in parallel, making it useful for robust and efficient online learning systems [27,28]. To the best of our knowledge, we are the first to integrate online learning, AutoML, and SSL in this context.

Our method integrates online implementation, self-supervised data labeling, and AutoML to reduce cycle times and streamline the training process. The system collects real-time data during each assembly cycle and builds multiple learning models through AutoML. It then evaluates the models and selects the best-performing one to correct misalignments based on insights from corrective strategies. Additionally, the system includes an intelligent decision-making process that dynamically selects between using the trained model or corrective strategies, optimizing the performance in a fully automated manner, without human intervention.

3. Proposed Method

Our system automates the process by capturing data during each cycle. The recorded data, denoted as $\mathbf{D} = \{\hat{P}, \epsilon_{\text{in-hand}}\}$, include two key components: the initial estimation of the grasping pose, denoted as $\hat{P}$, and the total in-hand error, denoted as $\epsilon_{\text{in-hand}}$, both represented as Cartesian poses. The total in-hand error represents the cumulative error corrected by both the picking refinement strategy, denoted as $\epsilon_{\text{picking}}$, and the flexible

assembly strategy, denoted as $\epsilon_{\text{placing}}$. The corrective strategies are explained in detail in Section 4.

The self-supervision in our system is derived from this pipeline, as the system autonomously generates the ground truth data and corresponding labels for the training of machine learning models. Specifically, the learning model input is the initial estimated pose $\hat{P}$, while the output, serving as the ground truth, is the cumulative correction $\epsilon_{\text{in-hand}}$; the learning model data flow is illustrated in Figure 2. This self-labeling mechanism enables the system to operate without manual intervention, continuously collecting and integrating new data to improve the model dynamically and adapt to changing production conditions.

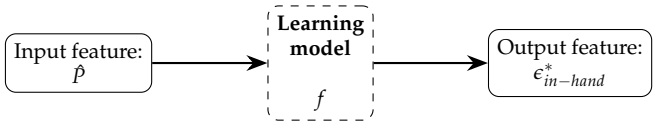


Figure 2. Model data flow.

Using Equation (1), the corrected grasping Cartesian pose, denoted as P, can be calculated. This grasping pose is designed to ensure that the object is centered in the gripper during the pick operation, allowing the assembly to be successfully completed using the predefined hole position.

$$P = \hat{P} - \epsilon_{\text{in-hand}} = \hat{P} - (\epsilon_{\text{picking}} + \epsilon_{\text{placing}}) \tag{1}$$

Model training occurs periodically after each assembly when the training set reaches a sufficient size. The input of the model, denoted as f, is the initial estimation of the grasping pose, $\hat{P}$. The training set, denoted as $\mathbf{D}_n$, is composed of several $\mathbf{D} = \{\hat{P}, \epsilon_{\text{in-hand}}\}$ acquired during each cycle in a self-supervised manner. The model output is the predicted total in-hand error $(\epsilon^*_{in\text{-}hand})$, represented as a Cartesian pose. Based on the predicted error and with the input data, $\hat{P}$, the grasping pose estimated by the model, denoted as P^*, can then be calculated using Equation (2). It is important to note that the grasping pose predicted by the model, P^*, is not equal to the corrected grasping pose obtained through corrective strategies, P. This discrepancy arises due to the associated error in the model's prediction.

$$P^* = \hat{P} - \epsilon^*_{in\text{-}hand} = \hat{P} - f(\hat{P}) = P^*(\hat{P}) \tag{2}$$

In this case, due to the online nature of the system, instead of training a single model, we employ an AutoML implementation where multiple learning models are trained with different hyperparameters and evaluated, and the best-performing one is selected for deployment. Therefore, in this section, when we refer to the selected model, we are referring to the model with the highest evaluation score from the set of trained models. More details about the models and scores used can be found in Section 4. Nevertheless, the usage of the selected model depends on two main requirements:

- The selected model's score, denoted as $\mathbf{S}_{model}$, must exceed a predefined threshold, denoted as T_{score};
- The input feature, $\hat{P}$, must not exceed a data drift threshold, denoted as $T_{datadrift}$.

Data drift evaluation involves monitoring the input data before the selected model's deployment to avoid applying the selected model to inputs that were not included during the training phase. With each new assembly cycle, a new estimated input data point, $\hat{P}$, is generated. This input is compared against the various estimated inputs stored in the training set. If the new input exceeds the data drift threshold $T_{\text{datadrift}}$, it indicates a potential drift and the selected model will not be deployed. Instead, corrective strategies will be employed, and the sample, $\mathbf{D}$, will be added to the training set. When both requirements are fulfilled, the selected model is deployed. The selected model is used to

predict the in-hand error of the pose $\hat{P}$, and, using Equation (2), the system moves to a corrected pose, P^*. This approach optimizes the performance by eliminating the need for corrective strategies. This process is illustrated in Figure 3, with further details provided in Section 4.

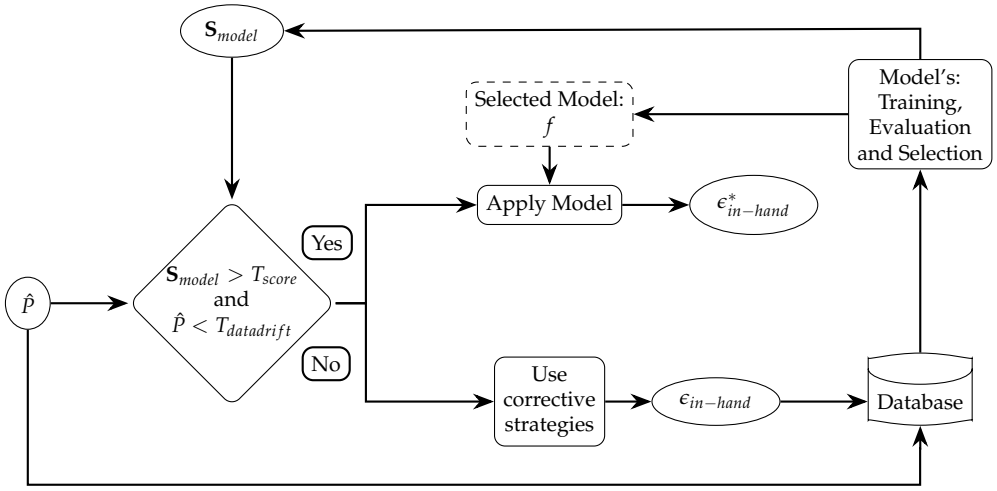


Figure 3. Decision logic for deployment of the selected model or corrective strategies based on thresholds. The system input is the estimated grasping pose $\hat{P}$, and the outputs are the total in-hand error $\epsilon_{\text{in-hand}}$ or the predicted error $\epsilon^*_{\text{in-hand}}$. The decision node determines whether to apply the selected model or use corrective strategies, with the results from corrective strategies saved in the database for model training.

4. Experiment

The purpose of this experiment is to evaluate the performance of our proposed method in a real-world scenario. The use case involves the assembly of a wheel onto a toy car, as illustrated in Figure 4. This scenario requires high accuracy for both assembly and grasping, with a tolerance level of 0.11 mm.

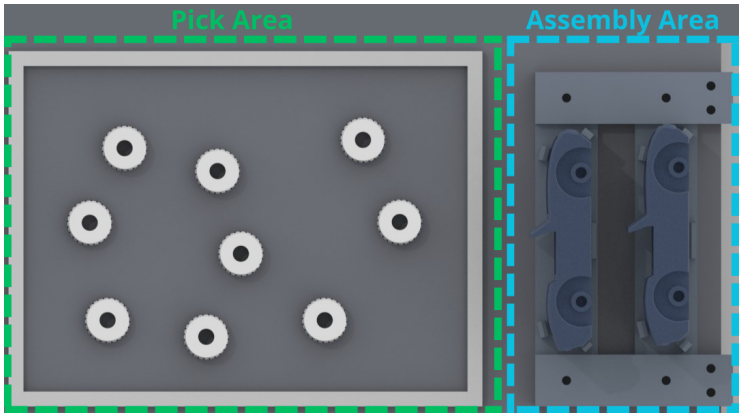


Figure 4. Representation of the use case area.

The setup includes an assembly area (a fixed workstation) and a pick area (with wheels in variable positions); see Figure 4. The mounted camera identifies the wheels via artificial vision and places them onto the car. The wheel hole positions are defined beforehand.

For our experiment, we employed a KUKA IIWA14 robotic arm, a collaborative robot with 7 degrees of freedom that provides impedance control. Attached to the arm is a PGC-140-50 parallel gripper, offering configurable gripping force and accurate control over object handling. Additionally, we integrated an Allied Vision Manta G-917B camera with a 12 mm lens, mounted on the robot's end-effector in an eye-in-hand configuration; see Figure 5.

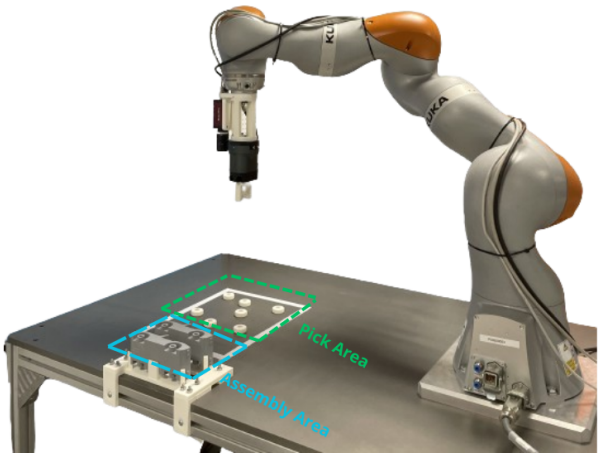


Figure 5. Experimental setup.

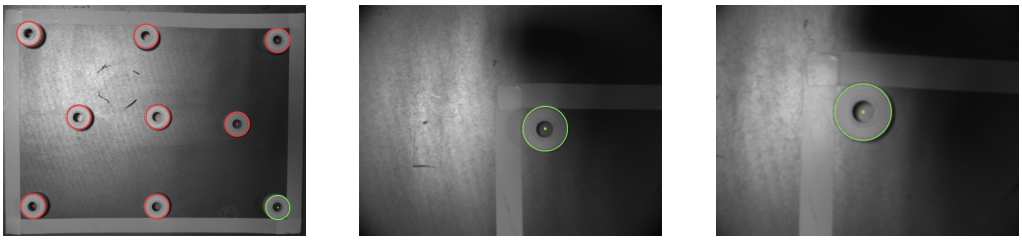
The intrinsic camera calibration was performed using the method proposed by Zhang et al. in [29]. This method yielded an intrinsic pixel error of 0.25 pixels for both axes. For the extrinsic calibration matrix, we tested all the methods discussed in the literature [17–22] and found that the separate methods provided the best results, particularly the Tsai method [17]. This aligns with the findings of [15], which state that separate methods perform better when prioritizing accuracy in translation over rotation. In our case, we prioritized accuracy in translation as the tool's orientation was predefined to be perpendicular to the table. Consequently, the Tsai method [17] was selected for its superior accuracy in our experiment.

4.1. Definition

The cycle begins with the mounted camera capturing an image of the pick area. From this image, the centers of the wheels are identified using the Hough transformation, proposed by Hough et al. in [30]. The detected pixel coordinates are then translated into spatial coordinates by leveraging the known distance to the table and the intrinsic camera matrix. Subsequently, these spatial coordinates are transformed from the optical frame to the robot's base frame using the eye-in-hand transformation matrix. The robot's tool orientation is fixed to be perpendicular to the table, resulting in an estimated grasping Cartesian pose, denoted as $\hat{P}$. For this use case, since z and the orientation are constant, the Cartesian pose parametrization is reduced from (x, y, z, a, b, c), where (x, y, z) represent the translational component and (a,b,c) the rotation component as simply (x, y) values.

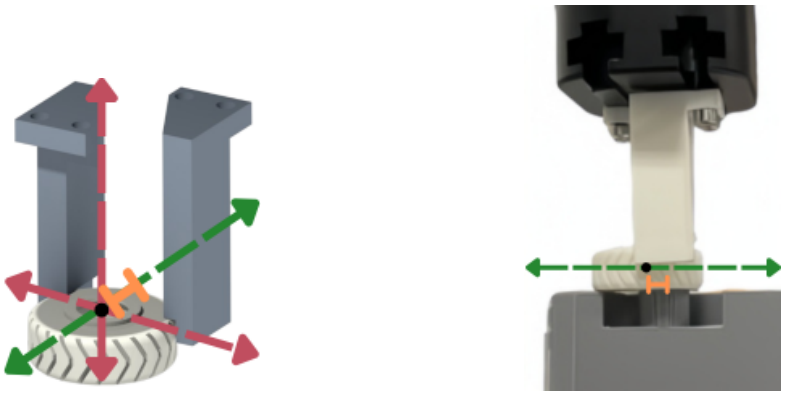
However, significant perspective errors can arise due to the distance between the wheel and the camera, which hinder the system's ability to grasp the wheel. The eye-in-hand configuration facilitates a picking refinement strategy that involves detecting the wheel, progressively approaching it, centering the camera, and refining the wheel's position, as illustrated in Figure 6. Through this process, we obtain an estimated grasping pose, $\hat{P}_i$, where i represents the number of refinement steps taken, ensuring sufficient accuracy for successful wheel picking.

Even if the piece is picked, the accuracy is compromised by the remaining in-hand error. For cases involving parallel grippers picking shape-symmetric objects, the stroke axis of the fingers being perpendicular to the object's symmetry axis acts as a centering mechanism, thereby limiting the in-hand error to one degree of freedom; see Figure 7. This error significantly impacts subsequent assembly operations, causing an offset in the axis, as illustrated in Figure 7b.



(**a**) (**b**) (**c**)

Figure 6. Stages of the refinement process for wheel detection: (**a**) the initial capture, (**b**) an intermediate capture during progressive approach, and (**c**) the final capture where the wheel position is refined for high accuracy.



(**a**) (**b**)

Figure 7. Representation of degrees of freedom: (**a**) the degrees of freedom during picking, with the red axes fixed, green axes free and orange in-hand error, and (**b**) the degree of freedom during the placing task.

To address the residual errors, we implemented a heuristic that enhances the flexibility of the assembly process. This strategy is based on the flexible assembly strategies discussed in the literature [4–6,8,9]. Specifically, we adopted the heuristic from [6], which directs the robot to follow a Lissajous curve trajectory as a hole search route; see Figure 8. This method effectively adapts to the shape of the errors and optimizes the search along the degree of freedom axis while smoothly adjusting in the perpendicular direction. The need for movement in this perpendicular direction arises from the robot's operation in impedance mode, where the positioning accuracy may decrease. However, it is important to point out that both the picking refinement strategy and the flexible assembly strategy can lead to increased cycle times on the assembly line.

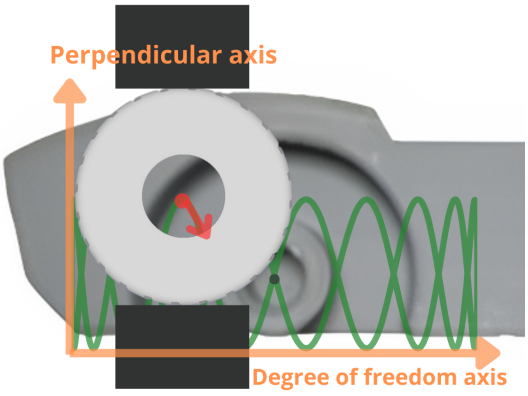


Figure 8. Lissajous search route curve.

4.2. Model-Based In-Hand Error Correction

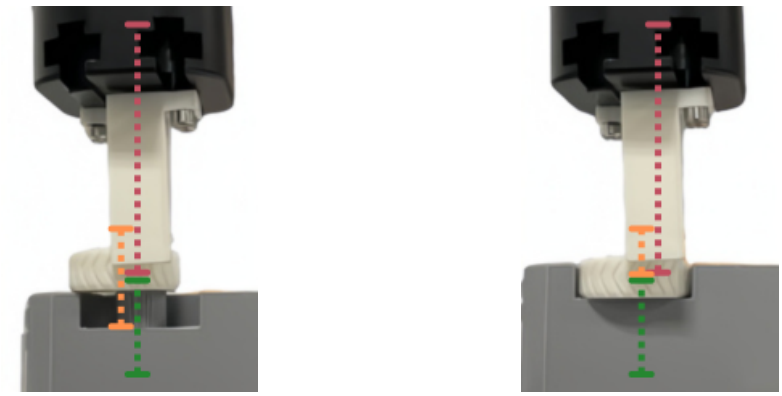
The proposed method requires recording a training set, consisting of multiple instances $\mathbf{D} = \{\hat{P}, \epsilon_{\text{in-hand}}\}$ obtained in each cycle, where $\hat{P}$ is the initial estimate of the grasping Cartesian pose and $\epsilon_{in-hand}$ is the in-hand total error, calculated as the sum of $\epsilon_{picking}$ and $\epsilon_{placing}$.

In this experiment, $\epsilon_{picking}$ is calculated in Equation (3). Here, $\hat{P}$ represents the estimated grasping pose from the initial image, while $\hat{P}_i$ denotes the estimate from the closest subsequent image. Figures 7a and 7b illustrate the captured images in these two positions, respectively.

$$\epsilon_{picking} = \hat{P} - \hat{P}_i \tag{3}$$

On the other hand, $\epsilon_{placing}$ is calculated using Equation (4). Here, $P_{Lissajous}$ indicates the robot's pose upon completing the peg-in-hole task using the Lissajous path, while P_{taught} is the intended pose for assembly, determined through teaching. Figure 9a,b illustrate these two positions, respectively.

$$\epsilon_{placing} = P_{Lissajous} - P_{taught} \tag{4}$$



(**a**) (**b**)

Figure 9. Poses for calculation of placing error: (**a**) the intended position P_{taught}, showing misalignment between the hole axis (green) and the wheel axis (orange), although the gripper axis (red) is aligned with the hole axis, and (**b**) the actual position $P_{Lissajous}$ achieved using the Lissajous path, where the hole axis (green) and wheel axis (orange) are aligned, but the gripper axis (red) is misaligned with the hole axis.

In each cycle, data from successful picking operations are collected and processed before being added to the training set. First, the samples, including both the estimated grasping Cartesian pose $\hat{P}$ and the in-hand error $\epsilon_{\text{in-hand}}$, are normalized and compared with the training set using the interquartile range (IQR) method, proposed by McGill et al. in [31], to filter out atypical values. Only samples that are not identified as atypical are included in the training set. To manage the growing database, a maximum sample limit is imposed. If this limit is exceeded, k-means clustering is employed to reduce the number of samples.

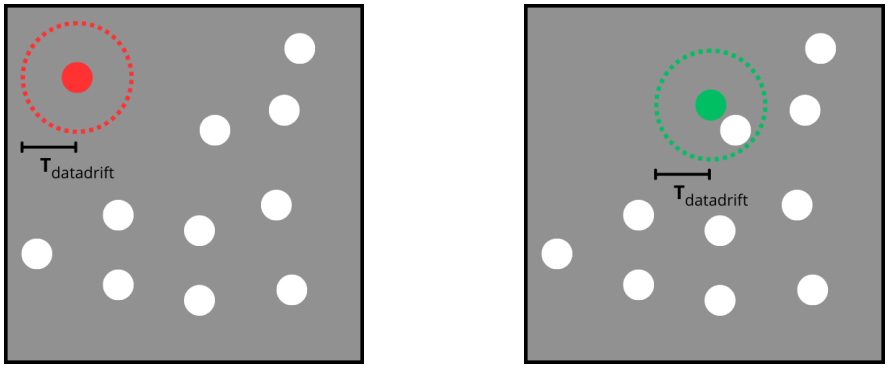
Due to the online nature of the system, instead of training a single model, we use an AutoML implementation with the PyCaret library [32]. PyCaret trains and evaluates multiple models in parallel, including linear models (linear regression, ridge regression), non-linear models (random forest, gradient boosting, support vector regression), and others, like quantile regression. Although PyCaret does not support neural networks, it is sufficient for the validation of the proposed pipeline. Since many PyCaret models (e.g., support vector machines, random forests, k-nearest neighbors) are univariate, we train separate

models for each dimension. In this experiment, the Cartesian pose is reduced to x and y coordinates, so we train one model for x and another for y.

For each dimension, all models available in PyCaret are trained separately. Each model is then evaluated using 10-fold cross-validation, providing the best regression model for each dimension, resulting in two selected models, one for each axis. PyCaret also provides a score vector, denoted as $\mathbf{S}_i = \{R^2, MAE\}$, for each selected model.

The selected models' deployment relies on the process outlined in the diagram shown in Figure 3. When a new cycle begins, the first grasping pose, $\hat{P}$, is estimated. At this stage, there are two options: either use the corrective strategies, which may take more time, or deploy the selected models.

For the selected models' deployment, their score vector, denoted as $\mathbf{S}_{model}$, is compared to a predetermined threshold vector, $\mathbf{T}_{score} = \{T_{R^2}, T_{MAE}\}$, to ensure its adequacy. For a selected model to be accepted, both metrics must surpass their respective thresholds, which means $\mathbf{S}_{R^2} >= T_{R^2}$ & $\mathbf{S}_{MAE} <= T_{MAE}$. Note that both the x and y selected models must surpass the predetermined threshold vector. Next, $\hat{P}$ is compared to the nearest sample in the training set; see Figure 10. If the Euclidean distance between $\hat{P}$ and the nearest sample exceeds the predefined threshold $T_{datadrift}$, the selected models are considered insufficient, and a new sample is collected to further explore that area. Conversely, if the distance does not exceed the threshold, the selected models are considered capable of explaining the $\epsilon_{in-hand}$ associated with that pose. In this case, the selected models meet the criteria to replace the existing corrective strategies.



(**a**) (**b**)

Figure 10. Explanation of the data drift threshold. The white circles represent the positions explored in the training set. (**a**) The input position exceeds the predefined threshold, indicating potential data drift and disabling the selected models' usage for these data. (**b**) The input position does not exceed the threshold, suggesting that the selected models remain valid.

4.3. Results

The experiment was conducted over 100 cycles. The demonstration can be seen in the corresponding video (demonstration video available at https://www.youtube.com/watch?v=5NOlWvdF94A, (accessed on 20 December 2024)). The accuracy threshold was set at $\mathbf{T}_{score} = \{0.9, 0.1\}$, while the data drift threshold was set at $T_{\text{threshold}} = 15$ cm, corresponding to the wheel's radius.

The system's performance was evaluated based on two key metrics: the cycle time reduction and the success rate. These metrics were selected to assess the method's ability to optimize the overall workflow and maintain reliable operation throughout the experiment.

The implementation of the model resulted in a significant reduction in the cycle time; see Figure 11. As the number of cycles increased, both the pick and assembly times demonstrated notable improvements. Specifically, the pick time decreased from 11 s to 7 s, while the assembly time was reduced from 14 s to 7 s, leading to a total cycle time improvement

of 45%, as the complete cycle time decreased from 25 s to 14 s. The system achieved a 100% success rate across all 100 cycles, ensuring that the wheel was successfully picked and correctly assembled in every attempt. This consistent performance highlights the method's reliability and effectiveness in maintaining high accuracy throughout the process.

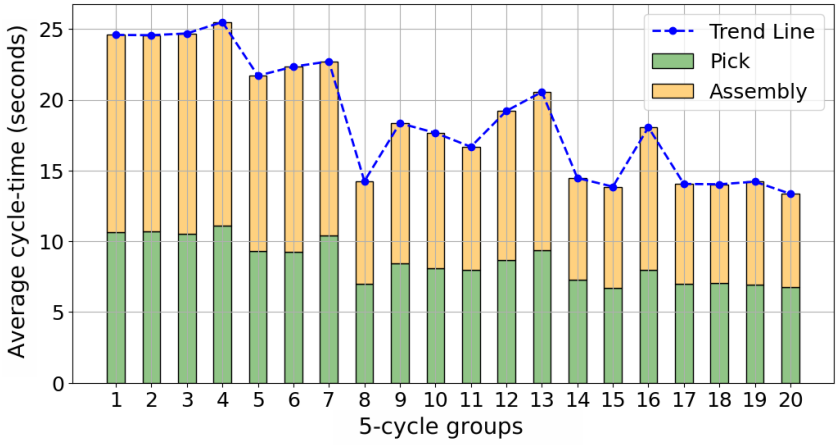


Figure 11. Cycle times over 100 cycles, grouped into 5-cycle segments. The vertical axis represents the average cycle time for each group of 5 cycles, while the horizontal axis indicates the group number. Each group divides the total cycle time into the pick time and assembly time. A trend line illustrates the reduction in cycle time as the experiment progresses.

The online implementation of the method was successful, as illustrated in Figure 12. After the initial eight samples, the models began their training and evaluation. However, these early samples were insufficient to train a model that could meet the $\mathbf{T}_{\text{score}}$. An additional 12 samples were needed to exceed the threshold. This aligns with the results shown in Figure 11. Until iterations 25 to 30 (group 5), the average cycle time remained stable and not optimized. From iteration 20, the selected models were deployed only when the input value, $\hat{P}$, met the $T_{\text{datadrift}}$. The first deployment occurred in iteration 25. This also corresponds to the trends in Figure 11 and demonstrates a decrease in the cycle time according to the selected models' usage shown in Figure 12. This method allowed the system to explore and adapt to the entire pick area, completing this exploration around cycle 80.

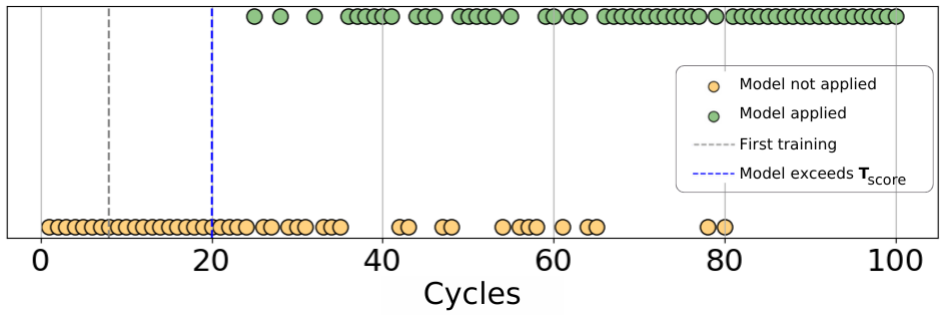


Figure 12. Decision-making as the experiment progresses.

Each time a new area was explored, a new sample was added to the training set, prompting the models to be retrained. This retraining process ensured that the selected models remained accurate and relevant as new conditions were encountered. Among all the trained models, linear regression consistently achieved the highest score in every training step, for both the x and y axes. Therefore, the deployed models in this experiment were linear regression models whenever it was necessary to utilize them

On the other hand, the model outputs, from both the x and y models, each had an associated error of 0.7 mm. This is significantly below the accuracy requirement for the use case, which is approximately 0.11 mm, ensuring consistently high accuracy in placement. For this reason, all assemblies were completed successfully, both when using the model and during the initial cycles without it.

5. Discussion

As shown in the previous sections, the system avoids production delays by autonomously developing and validating models during the operation cycle, unlike traditional offline implementations. This approach continuously collects data without interrupting production and dynamically decides whether to deploy the model or apply corrective strategies, ensuring real-time adaptability.

The proposed method was validated in the toy car wheel assembly use case. We evaluated the pipeline based on the cycle time reduction and success rate. The cycle time was significantly reduced, and the success rate was 100%, validating the self-supervised labeling, AutoML, and the decision-making system. The balance between the cycle time reduction and success rate is achieved through the T_{score} and $T_{\text{datadrift}}$ thresholds. These parameters ensure model deployment only when the accuracy requirements are met, while corrective strategies address situations where the model cannot perform reliably. By dynamically adjusting to each cycle's conditions, the system minimizes delays while maintaining a high success rate.

To guarantee that the proposed method is effective, the system's sensitivity to threshold parameters must be considered. The threshold $T_{\text{datadrift}}$ influences the system's responsiveness to data drift. Lower values of $T_{\text{datadrift}}$ increase the system's sensitivity, triggering model retraining more frequently and potentially resulting in longer cycle times if the model undergoes retraining when it is already adequately prepared for deployment. Conversely, a higher $T_{\text{datadrift}}$ threshold might allow the model to operate in unexplored zones, where it lacks sufficient accuracy and could negatively impact the grasping accuracy. In our use case, the threshold for $T_{\text{datadrift}}$ was set to 15 cm, corresponding to the radius of the wheel. This value was chosen based on the task's geometry and assumed to be large enough to capture significant data drift, ensuring that the model remained robust.

Additionally, the parameter $\mathbf{T}_{score}$ adjusts the values of the metrics that define the model, and these must be tuned depending on the specific use case. Using highly restrictive threshold values may lead to the model not reaching deployment, as it might never satisfy the defined objective. In contrast, less restrictive thresholds could compromise the performance. In our use case, we selected the metrics MAE and R^2. The MAE threshold was set to 0.1, which was smaller than the required accuracy for the task. We chose the MAE because it directly measures the average magnitude of the error, providing a clear and interpretable indication of the model performance. The R^2 threshold was set to 0.9, based on our experience with similar tasks, to ensure that the model explained at least 90% of the variance in the training data. R^2 was chosen because it indicates how well a model fits the data, providing a reliable measure of the model's explanatory power.

The model incorporates robustness mechanisms through the T_{score} and $T_{\text{datadrift}}$ thresholds, which determine whether corrective strategies should be applied. However, if the original conditions change, such as the partial calibration misalignment of the camera's extrinsic matrix or the displacement of the assembly area due to vibrations, the model adjusts its behavior using two mechanisms. In the picking stage, if the model fails, corrective strategies are applied and the ground truth sample is saved to progressively adapt the model to the new conditions. The second mechanism is activated in the placing stage, where, if the model fails, the flexible strategy is applied to ensure that the wheel is inserted

correctly. These adjustments allow the system to continue functioning and adapt to new conditions, although this may temporarily slow down the process.

The model's generalization depends on the type of errors that it addresses. While it generalizes well to similar pieces when correcting hardware-related errors, such as camera distortions or calibration inaccuracies, fine-tuning for specific object geometries limits its applicability to other scenarios without retraining. In our use case, the model adapts dynamically to both hardware and task-specific errors, showcasing flexibility in dynamic environments. However, retraining is necessary for different tasks due to variations in the error distributions.

Moreover, while our experiment demonstrated a linear distribution of in-hand errors, the proposed method should be capable of adapting to any error distribution, as long as it can be modeled by a machine learning model with sufficient data. In our experiment, the model selection was handled by AutoML through the PyCaret library, which evaluated 25 models, including nonlinear models like random forest, gradient boosting, and support vector regression. Although we did not use neural networks, as they typically require more data, our method is model-agnostic and could incorporate neural network models if necessary. The training pipeline does not make assumptions about the data distribution, meaning that all regression models are valid for training.

In addition to being model-agnostic, our proposed method is hardware-agnostic, meaning that it can be used with any robot, camera, and parallel finger gripper in an eye-in-hand configuration. The scalability of the method is determined by the type of process rather than the hardware used. While the workflow follows the same structure, it is adaptable to various picking refinement strategies or flexible placing strategies, not necessarily requiring both. This ensures that our method can be used across different manufacturing processes and tasks. This distinction highlights the versatility of our method, which, while validated in a specific application, is designed to be broadly applicable in future applications.

6. Conclusions and Future Work

In this work, we have presented a method to reduce the cycle time for accurate pick assembly operations in an online manner, without the need to delay the production process. The proposed method is self-supervised, meaning that it does not require any human intervention for labeling, as target values are automatically obtained during the process. For model training and validation, we employed AutoML and an intelligent decision-making system with thresholds, which ensures the model is continuously optimized and validated without interrupting production. By implementing this strategy, we achieve the efficiency, accuracy, and flexibility demanded in manufacturing automation, effectively addressing the challenges of grasping accuracy in vision-guided systems.

In future work, we aim to extend the method to more complex tasks such as pin-through-hole (PTH) pick-and-assembly tasks and improve its generalization and scalability. Additionally, while the online training has been conducted locally on a desktop computer, future work may explore its deployment on edge devices, which are better suited for industrial environments.

Author Contributions: Conceptualization, S.V., M.O. and X.L.; methodology, S.V. and M.O.; software, S.V.; validation, S.V. and M.O.; formal analysis, S.V. and M.O.; investigation, S.V. and M.O.; resources, M.O.; data curation, S.V.; writing—original draft preparation, S.V.; writing—review and editing, S.V., M.O. and X.L.; visualization, S.V.; supervision, M.O. and X.L.; project administration, M.O. and X.L.; funding acquisition, M.O. and X.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the dAIedge project (HORIZON-CL4-2022-HUMAN-02-02, Grant Agreement Number: 101120726).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: We would like to acknowledge the assistance of ChatGPT in refining the English grammar and linguistic clarity during the manuscript writing process.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SSL	Self-Supervised Learning
AutoML	Auto Machine Learning
DL	Deep Learning
RL	Reinforcement Learning
IQR	Interquartile Range
FIT	Force–Torque Sensor
R2	Coefficient of Determination
MAE	Mean Absolute Error

References

1. Smith, C.E.; Papanikolopoulos, N.P. Vision-guided robotic grasping: Issues and experiments. In Proceedings of the IEEE International Conference on Robotics and Automation, Minneapolis, MN, USA, 22–28 April 1996; IEEE: Piscataway, NJ, USA, 1996; Volume 4, pp. 3203–3208.
2. Shugurov, I.; Pavlov, I.; Zakharov, S.; Ilic, S. Multi-view object pose refinement with differentiable renderer. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2579–2586. [CrossRef]
3. Yuan, S.; Ge, Z.; Yang, L. Single-Camera Multi-View 6DoF pose estimation for robotic grasping. *Front. Neurorobotics* **2023**, *17*, 1136882. [CrossRef] [PubMed]
4. Park, H.; Park, J.; Lee, D.H.; Park, J.H.; Baeg, M.H.; Bae, J.H. Compliance-based robotic peg-in-hole assembly strategy without force feedback. *IEEE Trans. Ind. Electron.* **2017**, *64*, 6299–6309. [CrossRef]
5. Park, H.; Park, J.; Lee, D.H.; Park, J.H.; Bae, J.H. Compliant peg-in-hole assembly using partial spiral force trajectory with tilted peg posture. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4447–4454. [CrossRef]
6. Jokesch, M.; Suchỳ, J.; Winkler, A.; Fross, A.; Thomas, U. Generic algorithm for peg-in-hole assembly tasks for pin alignments with impedance controlled robots. In Proceedings of the Robot 2015: Second Iberian Robotics Conference: Advances in Robotics, Lisbon, Portugal, 19–21 November 2015; Springer: Berlin/Heidelberg, Germany, 2016; Volume 2, pp. 105–117.
7. Gibbons, O.; Albini, A.; Maiolino, P. A Tactile Feedback Insertion Strategy for Peg-in-Hole Tasks. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 10415–10421. [CrossRef]
8. Song, R.; Li, F.; Quan, W.; Yang, X.; Zhao, J. Skill learning for robotic assembly based on visual perspectives and force sensing. *Robot. Auton. Syst.* **2021**, *135*, 103651. [CrossRef]
9. Abdullah, M.W.; Roth, H.; Weyrich, M.; Wahrburg, J. An approach for peg-in-hole assembling using intuitive search algorithm based on human behavior and carried by sensors guided industrial robot. *IFAc-PapersOnLine* **2015**, *48*, 1476–1481. [CrossRef]
10. Huang, Y.; Zhang, X.; Chen, X.; Ota, J. Vision-guided peg-in-hole assembly by Baxter robot. *Adv. Mech. Eng.* **2017**, *9*, 1687814017748078. [CrossRef]
11. Almaghout, K.; Boby, R.A.; Othman, M.; Shaarawy, A.; Klimchik, A. Robotic pick and assembly using deep learning and hybrid vision/force control. In Proceedings of the 2021 International Conference "Nonlinearity, Information and Robotics" (NIR), Innopolis, Russia, 26–29 August 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
12. Schoettler, G.; Nair, A.; Luo, J.; Bahl, S.; Ojea, J.A.; Solowjow, E.; Levine, S. Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 5548–5555.

13. Yasutomi, A.Y.; Ichiwara, H.; Ito, H.; Mori, H.; Ogata, T. Visual Spatial Attention and Proprioceptive Data-Driven Reinforcement Learning for Robust Peg-in-Hole Task Under Variable Conditions. *IEEE Robot. Autom. Lett.* **2023**, *8*, 1834–1841. [CrossRef]
14. Smith, C.; Karayiannidis, Y.; Nalpantidis, L.; Gratal, X.; Qi, P.; Dimarogonas, D.V.; Kragic, D. Dual arm manipulation—A survey. *Robot. Auton. Syst.* **2012**, *60*, 1340–1353. [CrossRef]
15. Enebuse, I.; Ibrahim, B.K.K.; Foo, M.; Matharu, R.S.; Ahmed, H. Accuracy evaluation of hand-eye calibration techniques for vision-guided robots. *PLoS ONE* **2022**, *17*, e0273261. [CrossRef]
16. Qiu, S.; Wang, M.; Kermani, M.R. A new formulation for hand–eye calibrations as point-set matching. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 6490–6498. [CrossRef]
17. Tsai, R.Y.; Lenz, R.K. A new technique for fully autonomous and efficient 3 d robotics hand/eye calibration. *IEEE Trans. Robot. Autom.* **1989**, *5*, 345–358. [CrossRef]
18. Chou, J.C.; Kamel, M. Finding the position and orientation of a sensor on a robot manipulator using quaternions. *Int. J. Robot. Res.* **1991**, *10*, 240–254. [CrossRef]
19. Park, F.C.; Martin, B.J. Robot sensor calibration: Solving AX=XB on the Euclidean group. *IEEE Trans. Robot. Autom.* **1994**, *10*, 717–721. [CrossRef]
20. Daniilidis, K.; Bayro-Corrochano, E. The dual quaternion approach to hand-eye calibration. In Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria, 25–29 August 1996; IEEE: Piscataway, NJ, USA, 1996; Volume 1, pp. 318–322.
21. Lu, Y.C.; Chou, J.C. Eight-space quaternion approach for robotic hand-eye calibration. In Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century, Vancouver, BC, Canada, 22–25 October 1995; IEEE: Piscataway, NJ, USA, 1995; Volume 4, pp. 3316–3321.
22. Li, W.; Dong, M.; Lu, N.; Lou, X.; Sun, P. Simultaneous robot–world and hand–eye calibration without a calibration object. *Sensors* **2018**, *18*, 3949. [CrossRef]
23. Bohg, J.; Morales, A.; Asfour, T.; Kragic, D. Data-driven grasp synthesis—A survey. *IEEE Trans. Robot.* **2013**, *30*, 289–309. [CrossRef]
24. Miller, A.T.; Allen, P.K. Graspit! a versatile simulator for robotic grasping. *IEEE Robot. Autom. Mag.* **2004**, *11*, 110–122. [CrossRef]
25. Haugaard, R.L.; Buch, A.G.; Iversen, T.M. Self-supervised deep visual servoing for high precision peg-in-hole insertion. In Proceedings of the 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, 20–24 August 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 405–410.
26. Peng, G.; Ren, Z.; Wang, H.; Li, X.; Khyam, M.O. A self-supervised learning-based 6-DOF grasp planning method for manipulator. *IEEE Trans. Autom. Sci. Eng.* **2021**, *19*, 3639–3648. [CrossRef]
27. Baratchi, M.; Wang, C.; Limmer, S.; van Rijn, J.N.; Hoos, H.; Bäck, T.; Olhofer, M. Automated machine learning: Past, present and future. *Artif. Intell. Rev.* **2024**, *57*, 122. [CrossRef]
28. Chauhan, K.; Jani, S.; Thakkar, D.; Dave, R.; Bhatia, J.; Tanwar, S.; Obaidat, M.S. Automated Machine Learning: The New Wave of Machine Learning. In Proceedings of the 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, India, 5–7 March 2020; pp. 205–212. [CrossRef]
29. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
30. Hough, P.V. Method and Means for Recognizing Complex Patterns. US Patent 3,069,654, 18 December 1962.
31. McGill, R.; Tukey, J.W.; Larsen, W.A. Variations of box plots. *Am. Stat.* **1978**, *32*, 12–16. [CrossRef]
32. Khan, M.A.; Zia, M.F.; Faisal, A. PyCaret: An Open-Source, Low-Code Machine Learning Library in Python. 2024. https://www.pycaret.org/ (accessed on 28 October 2024).