Estimation of relative position between objects for robotic insertion of LDOs using classification learning methods

Sher Hazan

Estimation of relative position between objects for robotic insertion of LDOs using classification learning methods

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical
engineering

Sher Hazan

Submitted to the Senate
of the Technion — Israel Institute of Technology
Nisan 5784 Haifa May 2024

This research was carried out under the supervision of Prof.Anath Fisher,
in the Faculty of Mechanical Engineering.

The author of this thesis states that the research, including the collection, processing, and presentation of data, addressing and comparing to previous research, etc., was done entirely in an honest way, as expected from scientific research that is conducted according to the ethical standards of the academic world. Also, reporting the research and its results in this thesis was done in an honest and complete manner, according to the same standards.

Some results in this thesis have been published as articles by the author and research collaborators in conferences and journals during the research period, the most up-to-date versions of which being:

E. Cohen, S. Hazan, R. Schneor, A. Fischer, and M. Zacksenhouse, "Learned impedance policies facilitate robotic assembly in general direction and integration of visual sensor," Under Review, 2024.

S. Hazan, R. Schneor, and A. Fischer, "Estimation of relative position between objects for robotic insertion of LDOs using classification learning methods," in IMVC, Israel Machine Vision Conference, Poster presentation, 2024.

The Technion's funding of this research is hereby acknowledged.

Contents

List of Figures

List of Tables

Abstract

This work proposes a robust method for evaluating the relative position between both rigid and non-rigid objects in real-time assembly tasks using computer vision and learning methods.

The proposed approach fuses depth data with RGB images using a classification neural network (CNN) architecture. Several CNNs were tested and compared, including early, late, and parallel RGB-D fusion architectures.

In order to use classification CNN on a continuous problem, the scene space was divided into sub-areas as classes. The classification resulted in the prediction of probabilities of each class. Subsequently, these probabilities were combined using a linear interpolation technique, resulting in an accurate evaluation of the relative position between the objects.

For the validation process of the proposed method, three Peg-In-Hole tasks were defined, varying in their complexity. The tasks included: (1) a classic PID with rigid objects; (2) wiring a 1 mm thick non-rigid wire to a rigid connector; and (3) wiring a non-rigid medical pipe to a non-rigid connector.

An RGB-D realistic database was created for each task's CNN. The scenes were captured using a 3D camera in a real robotic cell at the Technion. Then, pre-processing and augmentation techniques were applied to the data. Finally, the CNNs were trained separately using the same setup and hyper-parameters to ensure comparable results.

The method was evaluated by embedding and testing it in the robotic insertion process based on the following iterative stages:

- Predicting the position probabilities.
- Calculating the relative position between the peg and the hole.
- Correcting the peg position.

Once the peg is aligned with the hole, the insertion process is performed. Furthermore, the real-world applicability and generalization of the method were validated by embedding it in an industrial-grade robotic cell without additional training.

The feasibility of the proposed method was demonstrated in several tasks, showing an automatic process with high performance. Consequently, it fits well into the industrial environment for tasks involving the insertion of linear deformable objects (LDO) within a robotic cell.

Abbreviations and Notations

2D	:	two Dimensional
3D	:	three Dimensional
RGB	:	Red, Green, Blue
RGB-D	:	Red, Green, Blue and Depth
LDO	:	Linear Deformable Object
PIH	:	Peg-In-Hole
AI	:	Artificial Intelligence
ANN	:	Artificial Neural Networks
ML	:	Machine Learning
DL	:	Deep Learning
RL	:	Reinforcement Learning
CNN	:	Convolutional Neural Networks
FC	:	Fully Connected
GT	:	Ground Truth
MAE	:	Mean Absolute Error
MSE	:	Mean Squared Error
ROI	:	Region of Interest
CAD	:	Computer-Aided Design
GD	:	Gradient Descent
SGD	:	Stochastic Gradient Descent
y	:	Ground truth value
$\hat{y}$	:	Estimated value
n	:	Number of samples
η	:	Learning rate
w	:	Weight
D	:	Domain
$P(x)$	:	Probability to x
R	:	Radius
θ	:	Angle
F	:	Force
$\nabla_x L$	:	Gradient of the loss with respective to x
L^x	:	The x loss funstion

Chapter 1

Introduction

Robotic integration in various industrial fields has long been a main goal, aiming to enhance production efficiency and reduce costs. One of the challenges in integrating robots into production lines, especially in assembly tasks is handling the uncertainty of the robotic environment. Moreover, assembly tasks such as inserting one part into another, often demand high precision. A common robotic insertion task is the Peg-In-Hole (PIH) task.

To address this challenge and make production more efficient, robots should have the ability to handle uncertainties in the assembly process. While existing methods have been successful to reduce position uncertainties during the assembly of rigid objects based on their 3D models, they fall short when dealing with non-rigid objects. Such objects not only lack the availability of 3D models but also introduce additional uncertainty due to their flexible nature. Perception of the robotic environment is needed to overcome the position uncertainties. For that, many sensors are available where cameras are becoming commonly used [1]. Recently, 3D camera technologies have evolved and becoming affordable. These technologies add depth information to the existing 2D data. The additional depth information enables the enhancement of 3D environment perception.

In this work, a visual sensor is proposed to be embedded in the robotic insertion pipeline of rigid and non-rigid objects in PIH tasks. The visual sensor utilizes 3D cameras to capture the 3D scene, analyze it and send the 3D perception to the robotic system. The analysis aims to extract information on the relative position between the peg and the hole in the 3D scene. The estimation of the relative position between the objects, proposed in this research, is based on computer vision and classification learning methods.

The research is performed and evaluated along with industry. Therefore, it should meet industrial speed and accuracy requirements and to be evaluated in a realistic robotic cell.

Chapter 2

Overview

This section will provide an overview of the research field and background information covered in this work.

The overview begins with the definition of 2D and 3D scenes, followed by a discussion on scene capturing and representation methods. Subsequently, it delves into the fundamental principles of deep learning, with a focus on convolutional neural networks for classification and regression. Additionally, fusion methods for depth and RGB data integration within these networks are reviewed. Lastly, the Peg-in-Hole (PIH) task is introduced, highlighting its complexities with linear deformable objects (LDOs). Various solution approaches, including Impedance Control and the use of vision sensors, are covered as well.

2.1 2D and 3D Scenes

Visual scenes, whether they depict urban environments or engineering objects, consist of spatial and visual details. Such scenes comprise multiple objects, each defined by specific attributes such as shape, texture, and color. Together, these elements create a comprehensive snapshot of the environment.

Although computer vision methods yield good results in analyzing 2D scenes, the lack of depth dimension may cause the scene's essence and spatial characteristics to be lost. Consequently, the task of understanding the spatial relationships between objects in the scene is impaired when relying solely on the 2D information. Therefore, using 3D scene depth data might avoid this problem.

2.1.1 Scenes Capturing

While capturing 2D scenes has become a simple task using standard cameras based on the pinhole model [2], capturing 3D scenes is more challenging both in terms of model and technology. In recent years, the use of 3D cameras increased due to the development of low-cost depth cameras such as Microsoft Kinect [3] and Intel RealSense [4]. These cameras are based on two main technologies: LiDAR cameras (Figure 2.1 left) and stereo cameras (Figure 2.1 right). LiDAR technology uses light-based sensors that send light pulses and measure the time it takes for light to reflect. Stereo camera technology mimics human binocular vision, utilizing two offset lenses to capture scenes. The depth information is extracted by comparing the disparity between the two images [2].

Figure 2.1: RealSense D405 stereo camera (left) and RealSense L515 LiDAR camera (right).

2.1.2 Scenes Representation

After capturing the scene, the data must be represented in a way that can be stored, processed, and analyzed efficiently. While 2D data representation is trivial (RGB), for 3D information there are several methods of data representation. The most common methods are RGB-D (Figure 2.2 left) and point clouds (Figure 2.2 right) due to their intuitive nature.

The RGB-D method combines traditional color images, represented as a red, green, and blue (RGB) image, with depth data as the fourth channel. This combined data is often displayed in two separate images, one for the color image and one for the depth data. Point clouds are collections of data points represented within a 3D coordinate system. Each point within this cloud signifies a specific location in space, marked by its x, y, and z coordinates, and associated with an RGB color. While point clouds offer a 3D view of the scene, they take up a lot of storage space compared to RGB-D and are less intuitive to use.

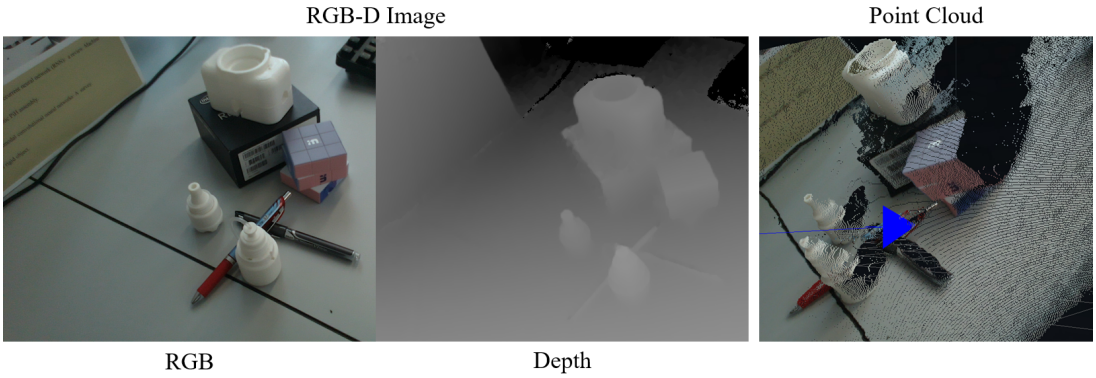


Figure 2.2: Side-by-side illustrating of RGB-D data (left) and point cloud representations (right).

2.2 Deep learning (DL) methods

Deep learning (DL) represents the cutting edge of machine learning (ML). These methods try to mimic the structure of the human brain to decode vast and complicated data.

In this section, the milestones of the development of deep learning will be presented, from the invention of the Perceptron [5] in the 1950's to convolution algorithms. In addition, evaluation and optimization methods in machine learning and deep learning will be explained in detail.

2.2.1 Evolution of Deep Learning (DL)

The perceptron model (Figure 2.3) which was first introduced by Rosenblatt in [5] stands as the foundational stone of artificial intelligence (AI), a simple binary classification algorithm that has sparked the imaginations of countless AI enthusiasts. Conceptualized in the 1950s, it sought to mimic a single neuron's behavior, adapting its weights based on input signals to yield an output. While elementary, the perceptron paved the way for more complex neural architectures.

The perceptron algorithm learns the values of the weights that are multiplied with the input feature vector to predict binary classification.

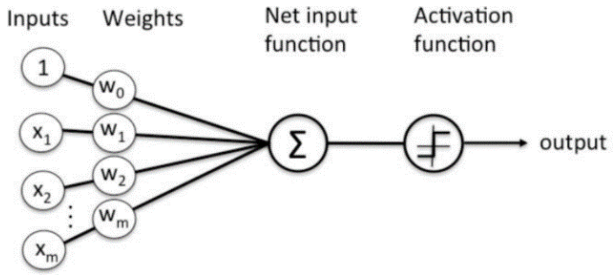


Figure 2.3: The Rosenblatt perceptron model.

The "Net input function" computes the sum of the multiplication of each input with its matching weight value (Equation 2.1).

$$z = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m \tag{2.1}$$

Then, an activation function is applied to this linear combination to determine the result of the classification. In Rosenblatt's case, the activation function is a threshold function also called the unit step function (Equation 2.2).

$$f(z) = \begin{cases} 1, & x > \theta \\ -1, & \text{otherwise} \end{cases} \tag{2.2}$$

Finally, the output value is the class label predicted by the unit step function. During the training, the weights are updated according to the learning rule (Equation 2.3).

$$w_i = w_i + \eta(\hat{y}_i - y_i)x_i \tag{2.3}$$

Where η is the learning rate, y_i is the true class label, and $\hat{y}_i$ is the output of the iteration.

Building on the foundation laid by the perceptron, Artificial Neural Networks (ANNs) represent another step in the evolution of this idea. ANN can be considered as a combination of perceptrons, which are connected to form a network. While a perceptron mimics the computational capacity of a single neuron, ANNs take inspiration from the biological neural networks of the human brain, aiming to replicate its vast connectivity and parallel processing capabilities for computational tasks.

The architecture of ANNs (Figure 2.4) consists of layers of connected nodes, or 'neurons'. These include:

- **Input Layer:** The gateway through which the network receives data. Each neuron here corresponds to a feature in the dataset.
- **Hidden Layers:** Located between the input and output layers, these intermediate layers consist of neurons that perform computations and transfer information from the input to the output layer. The presence of many hidden layers leads to the term "deep" in deep learning, as seen in deep neural networks (DNN).
- **Output Layer:** This layer produces the prediction or classification results of the network.

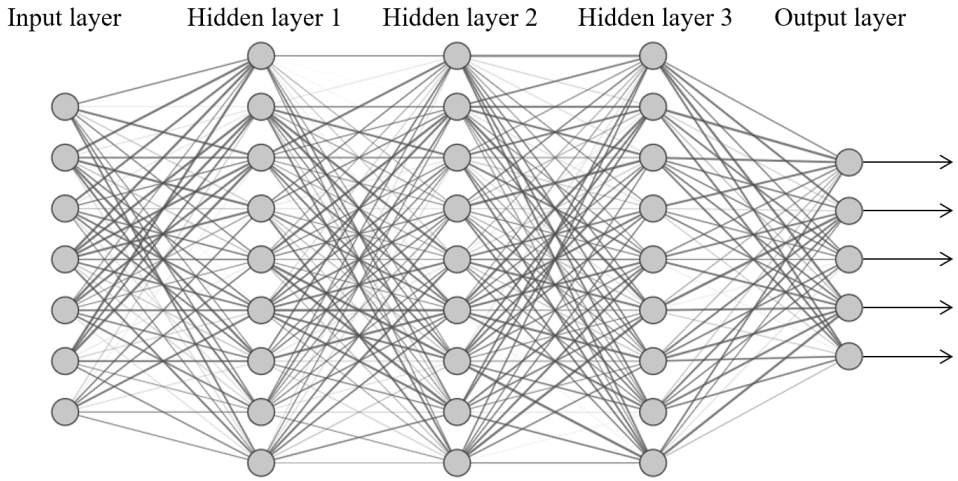


Figure 2.4: A diagram illustrating the architecture of an ANN.

To implement the data transition in the network, the neuron connections are made by weights. During the learning phase, these weights are updated. This is achieved using the back-propagation algorithm, combined with optimization strategies such as gradient descent.

Each neuron calculates a weighted sum of its inputs, similar to the perceptron model (Equation 2.1).

An activation function is then applied to this sum to produce the output of the neuron. Activation functions impart non-linearity to the network, allowing it to decode and represent complex patterns. The most common among the variety of activation functions are Sigmoid (Equation 2.4) and Rectified Linear Unit (ReLU) (Equation 2.5):

$$\text{Sigmoid:} \quad f(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

$$\text{ReLU:} \quad f(x) = \max(0, z) \tag{2.5}$$

During the network's training, the weights are fine-tuned to reduce the gap between the network's prediction and the actual labels. This operation is performed using optimization algorithms. Although the core concept is simple, it has practical complexities, especially in deeper networks. In the next chapter, these optimization techniques will be reviewed in detail.

In the vast landscape of neural network architectures, Convolutional Neural Networks (CNNs) are a field primarily concerned with processing visual data, such as images. Historically anchored by basic models like LeNet [6] (Figure 2.5) in the 90s. The rapid development of CNNs in the 21st century can be largely attributed to influential architectures such as AlexNet [7] and ResNet [8], both of which achieved remarkable performance on the ImageNet [9] Large Scale Visual Recognition Challenge. The ImageNet dataset, replete with millions of labeled images in thousands of categories, served as a benchmark to evaluate the capabilities of these CNN architectures.

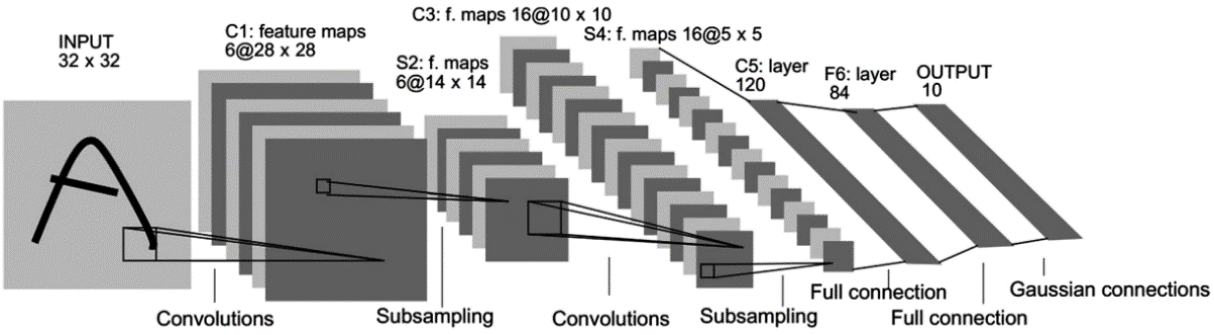


Figure 2.5: LeNet model diagram [6].

CNN, as the name suggests, revolves around the mathematical operation of "convolution". These networks demonstrate an ability to capture structured spatial hierarchies in images. Using a multilayer approach, they extract local features and create a collection of abstract image representations.

The architecture of CNNs (Figure 2.5) consists of several types of layers including:

- **Convolutional Layers:** The basis of CNN design, these layers use a set of learnable filters or kernels. Each kernel detects specific features in the input data, evolving from rudimentary patterns, like edges, in the first layers to complex structures in deeper layers. The convolution is performed by a kernel that passes over the image and performs linear operations on it as shown in figure 2.6.

- **Pooling Layers:** Typically embedded between convolutional layers, these layers aim to reduce spatial dimensionality while preserving relevant features. Pooling is done by dividing the image and taking the maximum value from each part as shown in figure 2.7.

- **Fully Connected Layers:** After extracting the features from the image using the convolution, they go through a fully connected ANN network (Figure 2.4) to make final predictions or classifications.

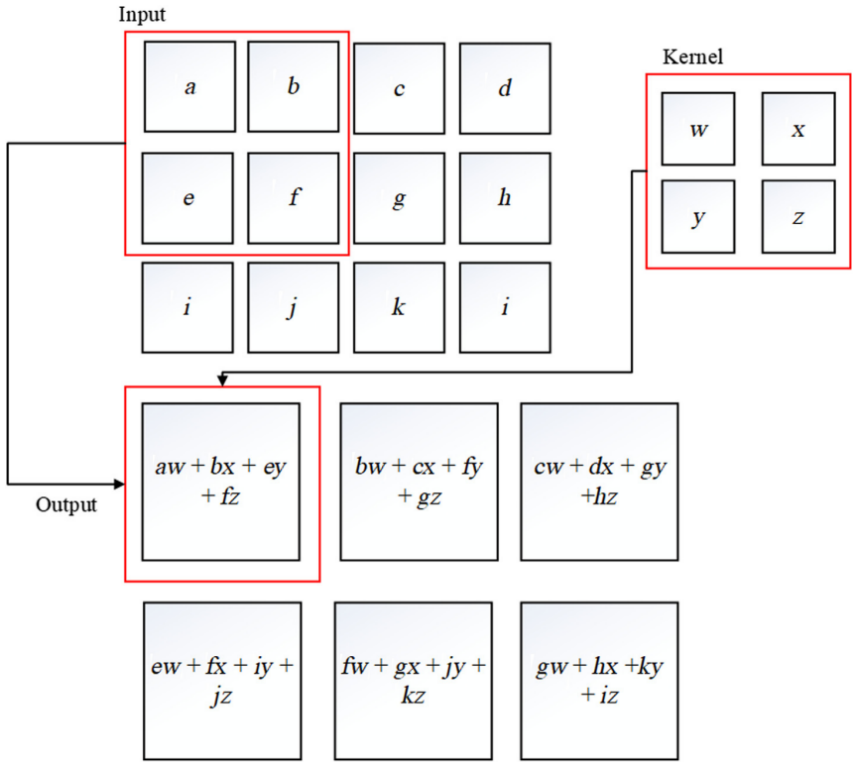


Figure 2.6: An example of a 2-D convolution [10].

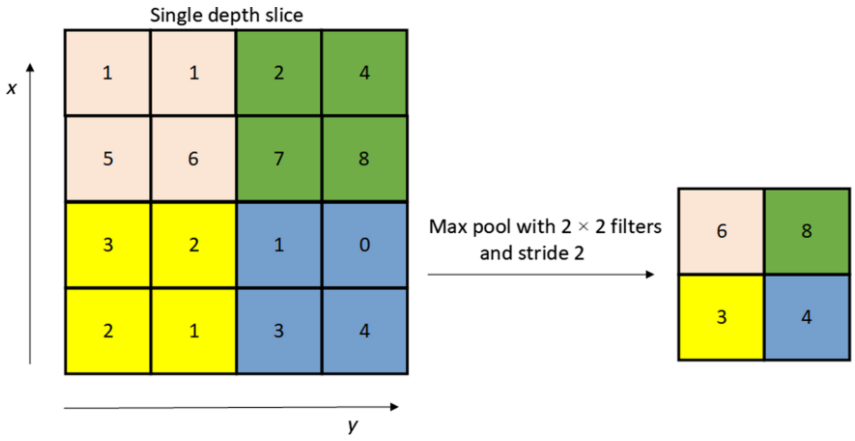


Figure 2.7: Schematic diagram of the max pooling layer [10].

A prominent feature of CNN is the way the weight is distributed. Unlike ANNs which can have unique weights for each input, CNNs reuse the same filter for all data, providing robustness to translational variation. This saves on the number of parameters, making CNNs more efficient and less susceptible to overfitting than ANNs.

2.2.2 CNN Evaluation and Training

After understanding the structure of the network, the different layers, and how they work comes the training phase. The training process is an iterative process that refines and adjusts the weights of the network, which leads to incremental improvements in its predictions. The training process includes three main components.

- **Loss Function:** The loss function is used as a measure of the prediction accuracy of the network. It quantifies the difference between the actual results and the network predictions. In fact, it's a mathematical way of expressing how 'off' our predictions are. By reducing this loss, we aim to improve the network's accuracy. There are many types of loss functions, each suitable for different types of tasks.

- **Optimization Algorithm:** After quantifying the quality of the network using the loss function, the optimization process must be performed in order to update the weights to improve performance. While several optimization algorithms exist, gradient descent (GD) stands out as the best-known and most widely used. It works iteratively, adjusting the weights in the direction that minimizes the loss.

  While the gradient descent (GD) algorithm lays the foundations, many variants such as stochastic gradient descent (SGD) and the Adam Optimizer offer diverse advantages, while addressing the limitations and challenges of the original method.

- **Learning Rate:** After finding the direction that minimizes the loss, the learning rate, η. This value determines the size of the steps taken during each optimization iteration. A high learning rate might lead the algorithm to overshoot the optimal point, while an exceedingly low rate can either cause exceedingly slow convergence or result in the algorithm becoming trapped in local minima. Fine-tuning the learning rate often proves crucial for achieving optimal network performance.

In the training process, a sample is passed through the network and the "distance" between the network prediction and the original sample labeling is quantified using the loss function. After which the optimization algorithm and the learning rate update the weights as shown in equation 2.6. The process repeats and in each iteration, the weights are updated so that the loss function decreases until the required accuracy is reached.

$$w = w - \eta \nabla_w L \tag{2.6}$$

Where w represents the weights, η is the learning rate, and $\nabla_w L$ is the gradient of the loss with respect to the weights.

2.3 CNN for Classification

The classification task deals with distinguishing and classifying data into predefined labels or classes.

Deep learning has shown remarkable abilities in classification tasks, where it uses multi-layer neural networks to process and classify complex data with great accuracy. Convolutional neural networks (CNNs) have become the cornerstone of image classification, a central field in visual recognition tasks, due to the ability to understand patterns in images and determine the relevant features for each class.

According to a comprehensive survey by Wang et al. [10], the architecture of CNNs applied in image classification not only laid the foundation for other visual recognition tasks such as object recognition and semantic segmentation but also underwent significant innovations and optimizations to improve the results. Popular architectures such as VGG [11] and ResNet [8] exemplify this, providing efficient feature extraction and classification. Once an image is processed through these multi-layer networks, a Softmax activation function is employed to transform the resulting prediction into a vector of probabilities that represents the probability of the image being classified into each of the classes. Figure 2.8 shows a general schematic of the CNN architecture for the classification task.

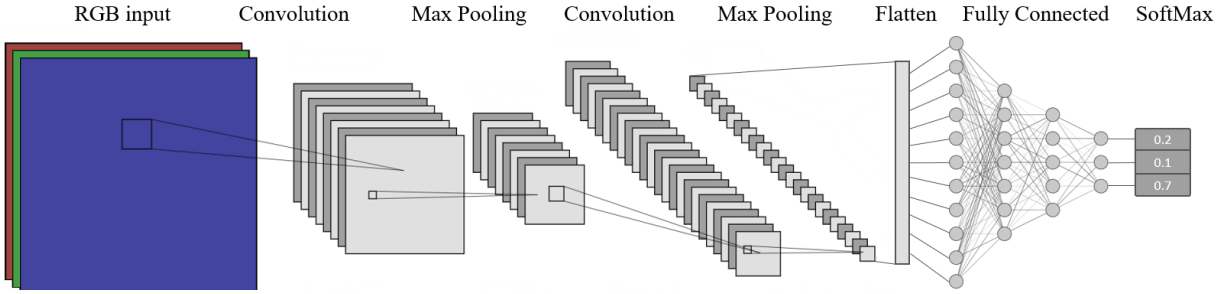


Figure 2.8: Schematic diagram of a Classification CNN.

The efficiency of a classification model is often evaluated based on the difference between its predicted class probabilities and the actual labels. Cross-entropy loss, also known as Log loss, stands out as the most common metric for the classification task. For binary classification, the Cross-entropy loss is calculated according to equation 2.7 where a lower Cross-Entropy value indicates a model with better predictive accuracy. An iterative optimization algorithm is activated in the training process which aims to minimize the loss and increase the accuracy of the network.

$$L^{CE} = \sum_{i=1}^{n} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right) \tag{2.7}$$

Where y_i represents the actual label (0 or 1), $\hat{y}_i$ represents the predicted probability of the instance belonging to class i, and n is the number of data points.

2.4 From Classification CNN to Regression CNN

Deep learning architectures offer the flexibility to adapt from classification tasks, which map input variables to discrete output categories, to regression tasks, which map input to continuous output values. The transition from a classification model to a regression model involves several adjustments. For example, the output layer, which often uses a SoftMax activation function in classification tasks, can be modified to include linear or sigmoid activations for regression. This layer usually consists of a single neuron that predicts a continuous output value (Figure 2.9).

The loss function should be adjusted as well. Unlike classification models, which typically minimize cross-entropy, regression models minimize loss functions such as mean squared error (MSE) (equation 2.8) or mean absolute error (MAE) (equation 2.9) to measure prediction accuracy.

$$L^{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.8}$$

$$L^{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.9}$$

Where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the number of data points.

These adaptation methods have been widely applied in recent years. For example, Stephane Lathuiliere et al. conducted a comprehensive analysis of deep regression techniques [12]. They utilized two popular CNN architectures, VGG-16 [11] and ResNet-50 [8], which were initially designed for classification tasks, to perform regression for several tasks. This transition from classification to regression involved careful selection of network optimizers, batch sizes, and appropriate loss functions. Their results demonstrated that when properly configured, these re-engineered architectures could achieve performance metrics approaching state-of-the-art levels.

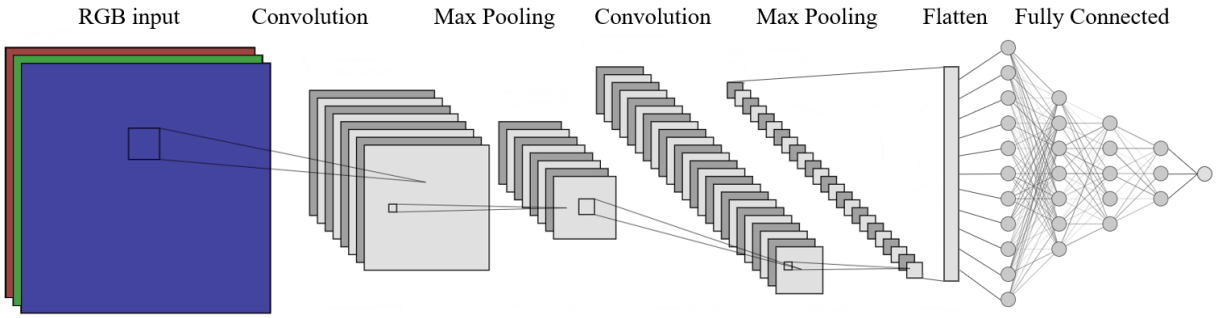


Figure 2.9: Schematic diagram illustrating the transition of a Classification CNN into a Regression Model.

2.5 RGB-D data fusion in CNNs

The effectiveness of deep learning (DL) architectures in 2D data processing has evolved due to the availability of large datasets for training. With the advent of affordable 3D data acquisition devices such as Microsoft Kinect [3] and RealSense [4], there has been an increase in the availability of public 3D datasets such as ShapeNets [13] and created interest in researching DL architectures based on 3D data. Researchers have taken different approaches to this challenge. Some have created new architectures specifically for 3D data, such as PointNet [14], while others have tried to improve existing architectures by fusing RGB and depth data [15]. Currently, there are two main RGB-D data fusion approaches in the field.

- **Early Fusion:** A trivial approach where RGB data and depth data are fused directly in the input layer, resulting in a multi-channel input tensor similar to a traditional RGB image but with additional channels for depth data. The fused input is processed by the network, making it easy to simultaneously extract features relevant to both RGB and depth data right from the start. Although the early fusion facilitates the fusion process and enables the extraction of mutual features at an early stage, it could become difficult due to the difference in scales between the RGB and the depth data.

- **Late Fusion:** This approach uses separate paths for RGB and depth data throughout most of the network architecture. For example, Ethel et al. [16] presented a dual-stream CNN for 3D object detection on RGB-D data. Two separate CNN architectures, identical in structure, are trained on RGB and depth channels independently. The outputs are then concatenated in a fully connected layer, with the SoftMax classifier performing the final class prediction. On the other hand, another approach [17] involves independent training of four CNNs, each for a separate channel, and finally all the networks are concatenated in a fully connected layer.

- **Parallel Fusion:** In this approach, RGB and depth data are processed independently by two distinct CNNs. Each CNN focuses on extracting features from its respective data type, leveraging their unique characteristics and patterns. After the feature extraction process, the outputs from both CNNs are merged either after the fully connected layers or later as post-process. This merging stage combines the extracted features from both RGB and depth data streams, allowing for a comprehensive representation of the input data.

2.6 Peg in Hole (PIH) task

The Peg-in-Hole (PIH) task (Figure 2.10) has long been a cornerstone in the field of robotics and serves as a crucial test to evaluate various automated systems in terms of accuracy, adaptability, and robustness. It should be noted that PIH tasks in different variations make up about 40% of all assembly operations, which emphasizes their significance in industrial settings. The task, which seems simple - inserting a peg into a hole - entails many complexities. These include the need for high repeatability, managing surface contact issues, and meeting tight assembly tolerances. To navigate these challenges, PIH missions require advanced perception and control strategies, particularly for achieving precise alignment, compensation for surface irregularities, and real-time adaptation to sensory feedback. Advances in sensor fusion technologies, which combine tactile and visual data, have helped increase the efficiency and reliability of PIH solutions [18].

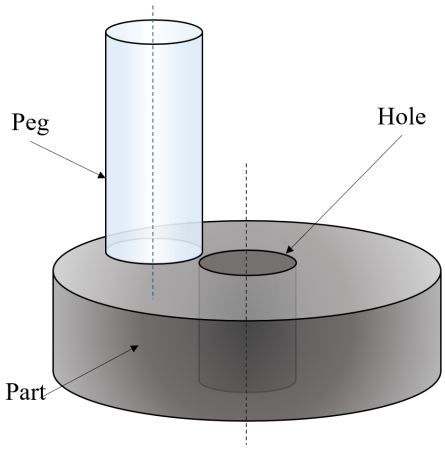


Figure 2.10: Illustration of the Peg-in-Hole task.

2.6.1 LDOs (Linear Deformable Objects) PIH

The inclusion of linear deformation objects (LDOs) in PIH tasks has added new dimensions of complexity and requires more advanced control strategies. Unlike rigid pegs, LDOs can change shape during the assembly process, which poses significant modeling, control, and performance challenges [19]. Accurate modeling of deformations, dynamic control of applied forces, and real-time adjustments for precise insertion are among the main obstacles. These issues are further complicated by external variables such as environmental conditions and the specific material properties of the LDOs. In response, advanced sensory feedback systems and machine learning algorithms are increasingly being used to navigate the complex challenges presented by LDOs in PIH missions. For example, the assembly of rigid pegs to deformable surfaces using learning methods was presented in [20].

2.6.2 Impedance Control and Reinforcement Learning (RL)

Impedance control, a concept first introduced by Hogan [21], is used as a transformative control strategy in the solutions of the PIH task. This method has drastically redefined how robotic systems interact with their environments by equipping the robot's end-gear with a Mass-Spring-Damper-like dynamic response (Figure 2.11), as explained in detail in [18]. This innovative approach provides six degrees of freedom to adapt to external contact forces and moments. Unlike traditional control mechanisms that focus exclusively on position or power errors, impedance control combines both parameters to create a nuanced control signal. This duality yields more compatible and adaptive interactions, especially essential when navigating irregular or imperfect surfaces. The main purpose of impedance control in PIH tasks is to maintain a predefined relationship between the applied force and the robot's motion, thus ensuring a stable and controlled insertion process. The impedance parameters can be tuned to suit specific tasks, materials, or conditions, making this control strategy incredibly versatile.

One of the tools for fine-tuning impedance parameters is Reinforcement Learning (RL) [22]. Unlike traditional machine learning methods, RL is trained using a trial and error method, which does not require predefined labels or structures. More insights into RL can be found in [23]. Subsequently, RL algorithms can tune the impedance parameters through interactions with the environment autonomously. This allows the robotic system to dynamically adapt to complex conditions, achieving unprecedented levels of adaptability and efficiency .

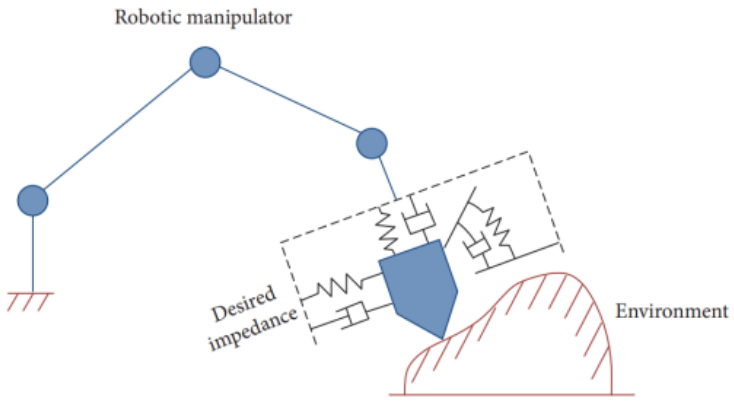


Figure 2.11: Illustration of impedance control for a robot in contact with its environment [24].

2.6.3 Visual Sensor

Visual sensors have become significant in PIH tasks, serving as the "eyes" of the automated systems. They play a central role in improving the system's ability to adapt to changing conditions and handle complex tasks. While impedance control is difficult in the field of flexible bodies due to the coupling of forces, visual sensors have proven to be effective in these tasks [25], [26].

Visual sensors allow robots to better perceive their operating environment. Concerning the PIH tasks, assisting them in accurately aligning pegs with holes even when dealing with irregular or imperfect surfaces. Advances in the use of CAD models and visual sensors have been particularly effective in handling complex shaped objects, further reinforcing the importance of visual sensors in PIH tasks [27], [28].

Chapter 3

Problem statement

Our research focuses on estimating the relative position between objects for Peg in Hole (PIH) tasks in real industrial environments, with a focus on linear deformable objects (LDO), using classification learning methods.

The insertion of rigid bodies using robotic arms has been well-researched, leading to numerous solutions. However, applying these methods to Linear Deformable Objects (LDOs) has proven challenging, with unsatisfactory results. This research aims to estimate the relative position between objects by employing learning-based classification algorithms for precise robotic PIH manipulations in a real industrial environment.

Figure 3.1: Camera setup.

In addition to our primary research aim, the following goals were established due to practical industrial requirements:

- **Real time Operation:** 30 [FPS].
- **Accuracy:** 1 [mm] error in the radius and 10 [degrees] error in the angle.
- **Robustness:** To environment changes and camera viewpoints.

Chapter 4

Approach

Our approach was to create a visual sensor for estimation of the relative position between objects and embed it in the insertion pipeline in a real robotic cell. The pipeline includes three main processes illustrated in Figure 4.1.

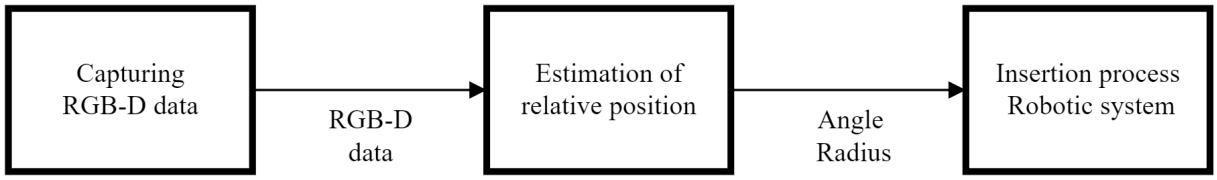


Figure 4.1: The pipeline of the insertion tasks with the embedded estimation method.

Capturing RGB-D data: The RGB-D data is captured in a robotic cell using two 3D cameras with different depth-capturing technologies. Both cameras were mounted on the robotic arm gripper focusing on the tip of the inserted object.

Estimating relative position: In order to apply classification methods to continuous tasks, the problem required discretization. This was accomplished by dividing the scene space into sub-areas as classes. Following the discretization of the task, a CNN is utilized to predict a vector representing the probability of the object being in each of the sub-areas. The CNN architecture is based on the ResNet-18 network due to its capability for real-time running, requires a small dataset for training, suitability for both classification and regression tasks, and ease of adaptation for RGB-D data. Finally, a linear interpolation is utilized to estimate continuous values for the relative position between the objects based on the predicted probabilities vector.

Insertion process: The actions of the robotic system are determined based on the estimated relative position.

A set of PIH test cases incorporating LDOs was defined in order to evaluate the process capability. The tasks included: the classical PIH with rigid bodies (Figure 4.2), wiring a non-rigid electric wire to a rigid connector (Figure 4.3), and inserting a non-rigid medical pipe into a non-rigid connector (Figure 4.4).

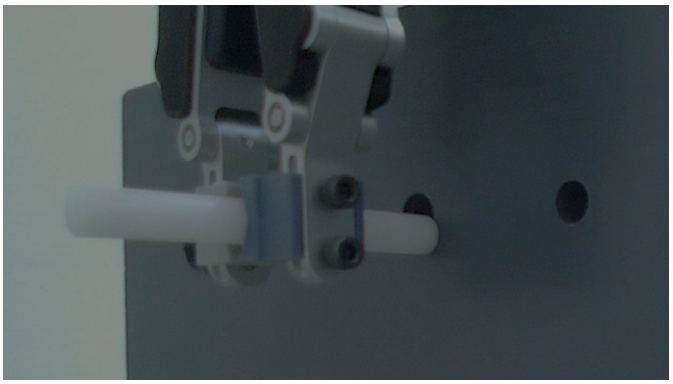


Figure 4.2: Classic Peg in Hole task.

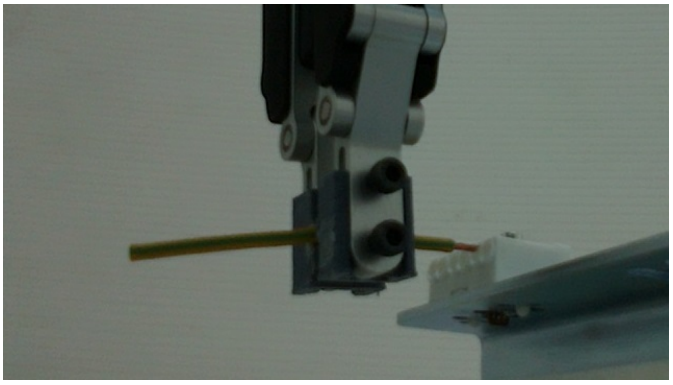


Figure 4.3: Wiring an electric wire task.

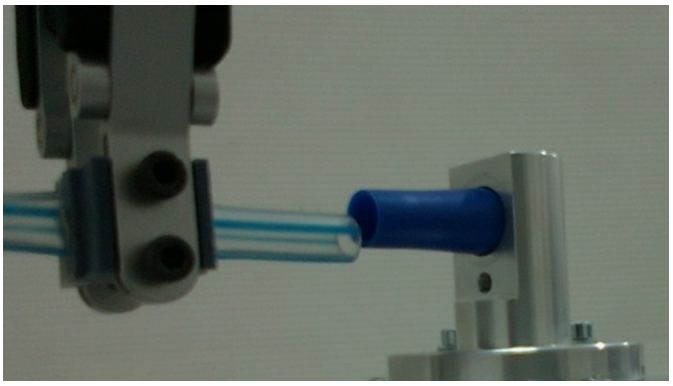


Figure 4.4: Insertion of medical pipe into a connector task.

Chapter 5

Implementation

This chapter delves into the implementation of our approach including: (i) the training process; (ii) the estimation of relative position process; (iii) the PIH insertion process.

5.1 Training process

The training process of the CNN models, which will later be utilized to predict the relative positions between objects in the various tasks, includes three main steps, shown schematically in Figure 5.1:

Capturing realistic 3D dataset: Capturing a realistic 3D scene for each task and camera combination.

Data pre-processing and augmentation: Pre-processing the data to expand the dataset and utilizing augmentation to enhance generalization.

Training the CNN models: Conducting CNNs training for each task-camera combination.

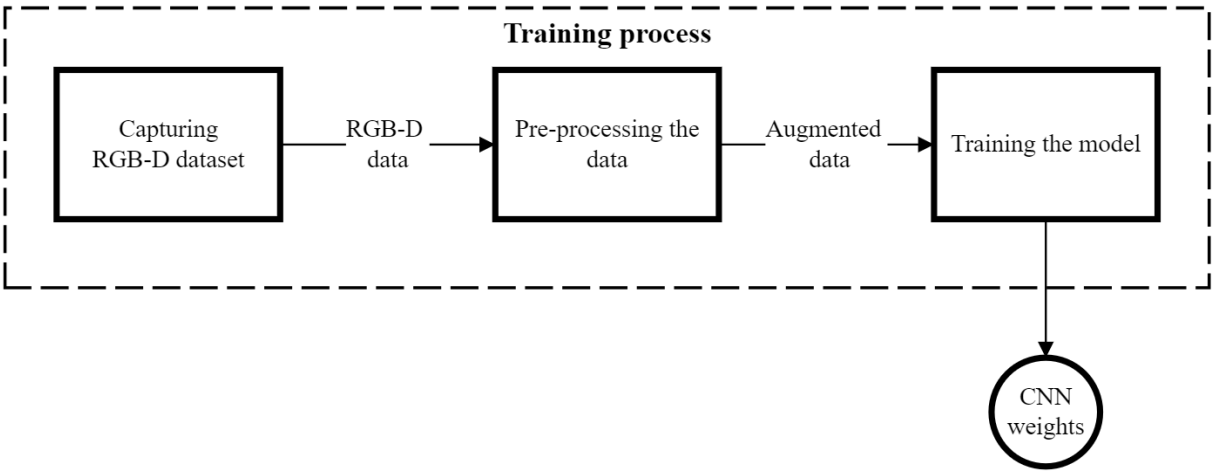


Figure 5.1: Training process scheme blocks.

5.1.1 Capturing realistic 3D dataset

The realistic 3D scenes were captured in the robotic cell using two RGB-D cameras with different depth capturing technologies, a stereo camera (Realsense 405) and a LiDAR camera (Realsense 515). Both cameras were mounted on the gripper of the robotic arm focusing on the tip of the inserted peg.

In order to focus on the region of interest (ROI) in the depth data, a camera configuration was set up which captures the relevant depth range for the task.

For each task, 2500 3D scenes (comprising RGB images depth data) were captured using both cameras. An example of the RGB-D data captured by each of the cameras can be seen in Figure 5.2.

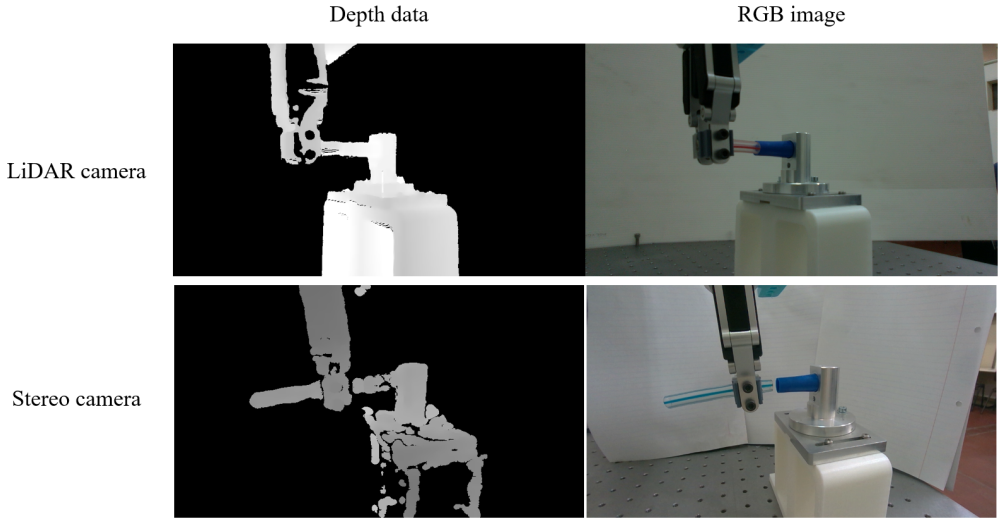


Figure 5.2: Samples of depth data and RGB images obtained from both cameras.

5.1.2 Data pre-processing and augmentation

In order to expand the dataset, pre-process procedure was implemented. Patches of various sizes and shapes were extracted from the RGB-D data. To ensure the patches would be extracted without losing information, each image was first centered based on the region of interest (ROI). An example of centered data is shown in Figure 5.3. From each 3D scene three unique scenes were extracted resulting in 7,500 scenes in each dataset. Samples of patches, extracted from the same 3D scene, are presented in Figure 5.4.

Additionally, to improve the generalization of the models, various augmentations were applied to the training set during the training process including color, lighting and translation augmentation. An illustration of augmented data is shown in Figure 5.5. In order to ensure the validity of the training process on the original data, no augmentations were applied to the validation set.

5.1.3 Training the CNN models

To ensure comparable results, all CNNs were trained using identical setup and hyperparameters. A batch size of 64 images and an adaptive learning rate initialized to 1e-4 were employed, utilizing

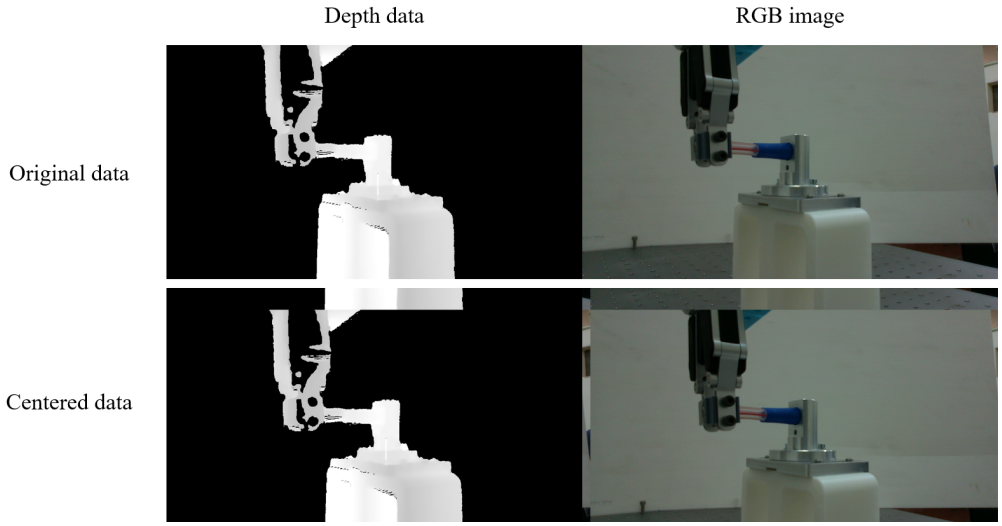


Figure 5.3: Examples of results achieved by the centering algorithm.

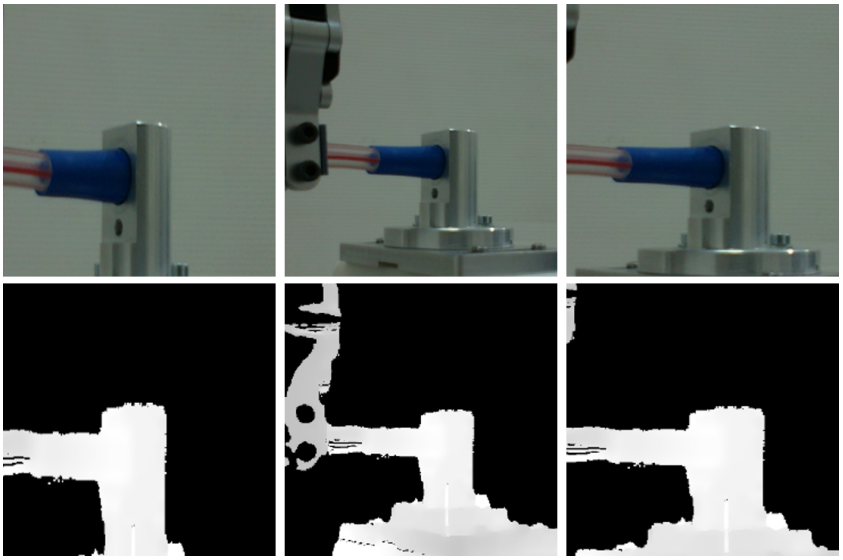


Figure 5.4: Samples of patches extracted from the same RGB image and depth data.

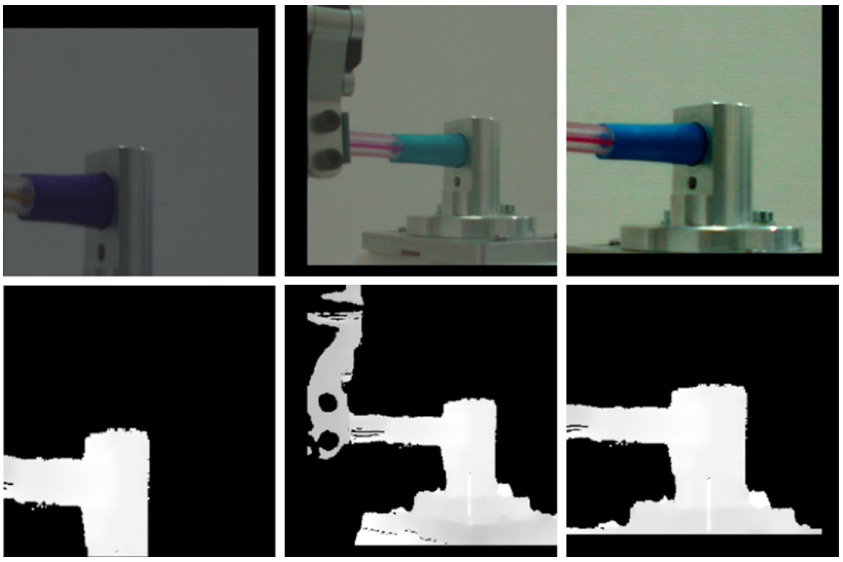


Figure 5.5: Same samples from Figure 5.4 after augmentations.

the Adam optimizer along with MSE Loss for regression or Cross-entropy loss for classification. Each model Passed through 25 epochs during the training.

Observing the training process performance and activation map revealed that the network focused better on images with a flat background than on images with a noisy background.

Therefore, a method for filtering the background of the RGB images using the depth data was proposed as part of the data pre-processing. The depth data was utilized to generate a binary mask, which was then employed to filter the background from the RGB image, as illustrated in Figure 5.6. As can be seen in the figure, the background filtering for the LiDAR camera presents promising results while for the stereo camera a loss of relevant RGB data is shown due to the noisy depth data.

Consequently, in order to compare the performance using both cameras, the 3D scenes were captured against a flat background to simulate the filtering. The filtering pre-process efficiency is evaluated using the LiDAR camera.

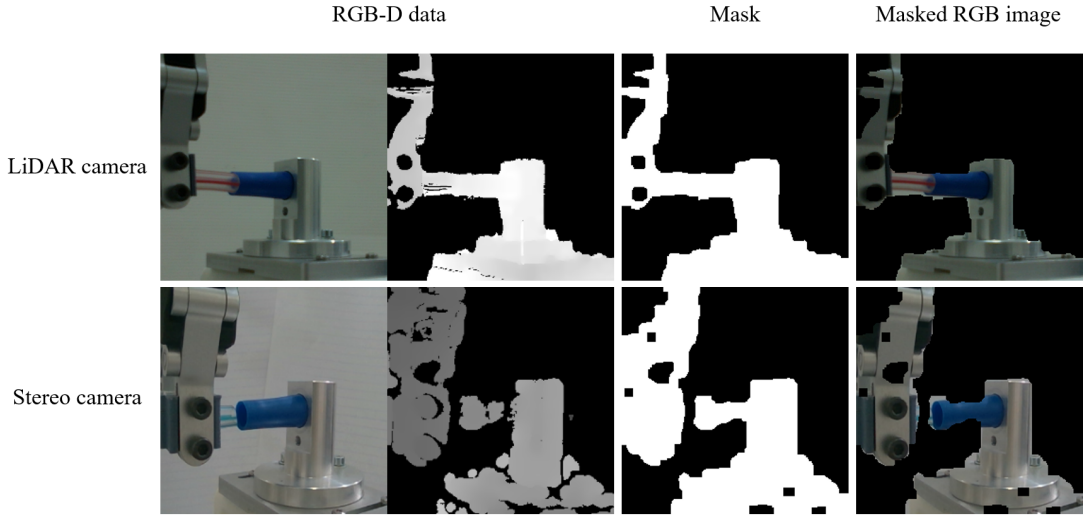


Figure 5.6: The process of filtering the background from the RGB image in both cameras.

5.2 Estimation of relative position process

The trained CNN models are utilized to predict the relative positions between objects in the various tasks. The relative position estimation and its application in the insertion process are shown schematically in Figure 5.7. This process consists of several steps:

Capturing 3D scene: The 3D scene is captured using a depth camera mounted on the robotic arm. The scene is represented as an RGB image and depth data.

Relative Position Classification: The trained classification CNN model is used to predict the probabilities vector of the relative position.

Relative position extraction: Perform a linear interpolation on the probabilities vector to extract the relative position in terms of angle and radius.

PIH insertion process: The robotic arm adjusts the peg position based on the estimated relative radius and angle. The entire process is iteratively performed until it meets the insertion conditions.

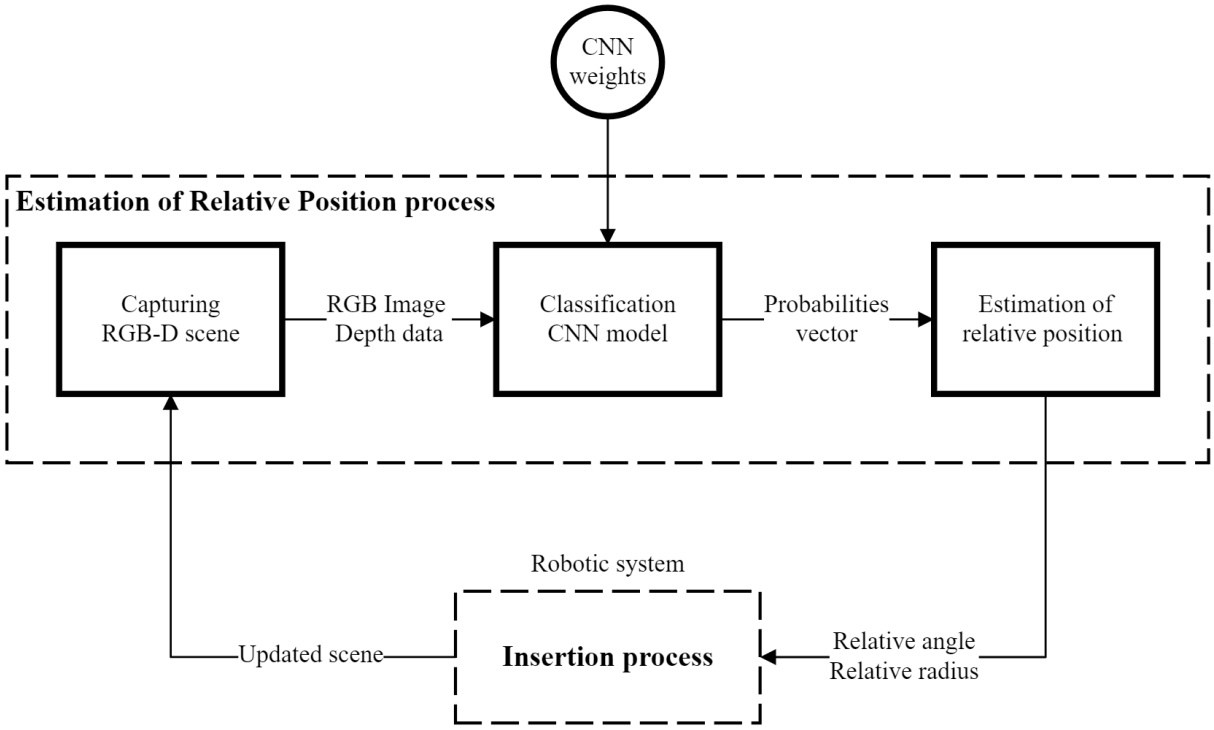


Figure 5.7: Estimation of relative position and Insertion process scheme blocks.

5.2.1 Capturing 3D scene

The process of capturing the RGB-D scene is carried out in real time at a rate of 30 frames per second via the camera mounted on the robotic arm. The RGB-D data is resized and cropped to a 240x240 patch to match the required input into the trained CNN model.

5.2.2 Relative position classification

The relative position classification is performed using several different CNN models. The models differ in their training dataset according to the task-camera combination and their architecture. Several CNN architectures were designed with different RGB-D fusing methods as well as for separate RGB and depth data as baseline. Each architecture is built based on Resnet-18 [8] without pre-training. The model's output is a probability vector, containing the likelihood of the input belonging to each class.

- **Separate RGB images and depth data architectures:** These architectures serve as the baseline for our analysis. Two separate models were implemented: (i) the RGB architecture, a standard ResNet-18 architecture; (ii) the depth architecture, where the input channels were reduced to a single channel to accommodate the data.
- **Early fusion architecture:** In this fusion approach, RGB images are concatenated with the corresponding depth data. This fused data, consisting of four channels (RGB-D), is then fed into the CNN, as illustrated in Figure 5.8.
- **Late fusion architecture:** In this fusion approach, two separate ResNet-18 CNNs, one for RGB images and one for depth data, are implemented. The features obtained from the convolutional layers in each CNN are concatenated before reaching the fully connected layer. Then this concatenated feature vector is inserted into one fully connected layer, as illustrated in Figure 5.9.
- **Parallel fusion architecture:** In this fusion approach, two separate ResNet-18 CNNs, one for RGB images and one for depth data, are implemented. The outputs of the fully connected layers are summed and normalized to obtain a combined probability vector, as illustrated in Figure 5.10.

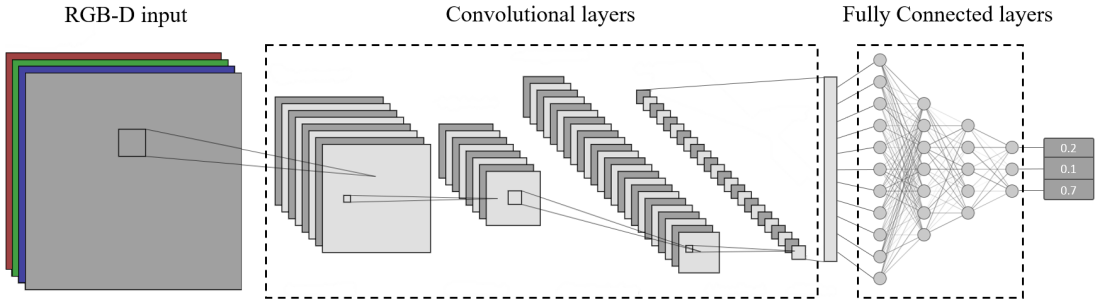


Figure 5.8: Simple schematic diagram of Early Fusion architecture.

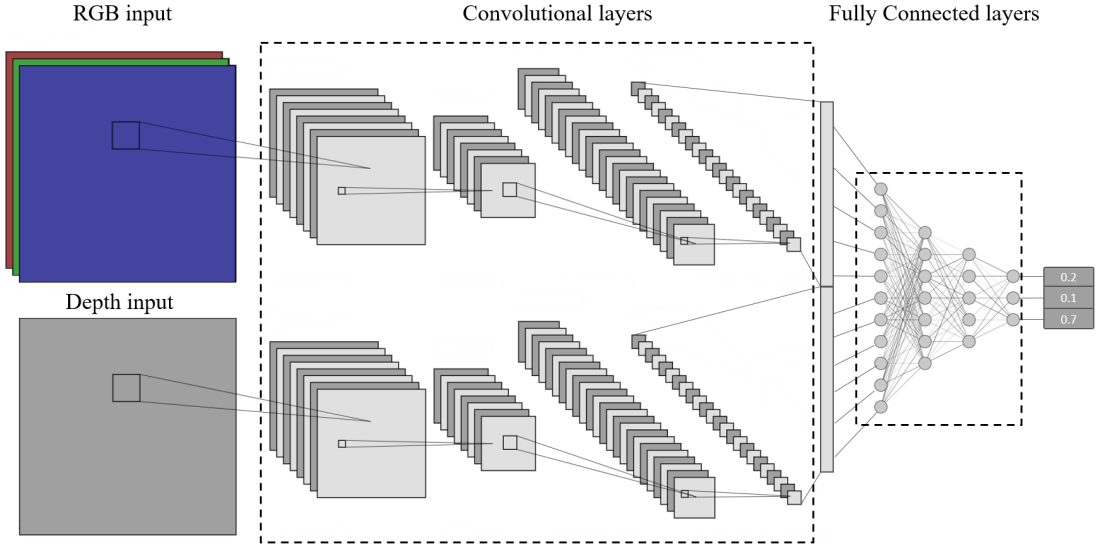


Figure 5.9: Simple schematic diagram of Late Fusion architecture.

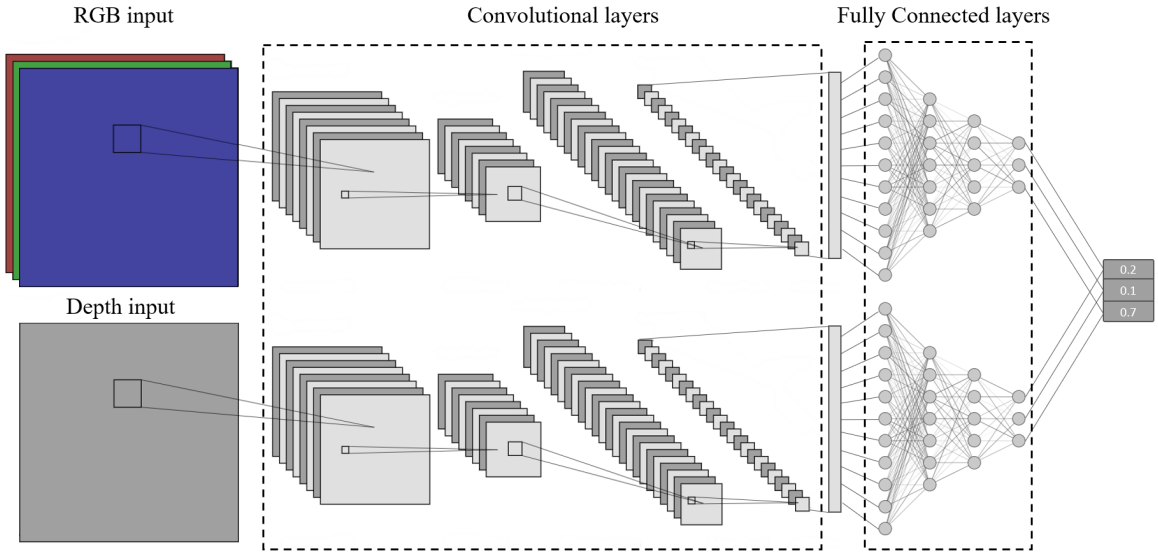


Figure 5.10: Simple schematic diagram of Parallel Fusion architecture.

5.2.3 Relative position extraction

The continuous relative position represented by the relative angle ($\hat{\theta}$) and relative radius ($\hat{R}$) was estimated using a linear interpolation applied on the predicted probability vector. This interpolation process involves thresholding negligible values to suppress noise, normalizing the filtered probability vector, and calculating the mean value across all classes based on the probability vector (Equation 5.1).

$$\hat{X} = \sum_{i=1}^{n} \frac{D}{n} \left(i - \frac{1}{2} \right) P(x_i) \tag{5.1}$$

Where D is the domain, n is the number of classes, and $P(x_i)$ is the probability for class i.

To determine the optimal number of classes (n), an experiment was conducted where the scene space was divided into various numbers of classes, and the network's performance was evaluated. It was observed that dividing the space into $n = 16$ classes yielded the best results for our task and dataset size, as illustrated in Figure 5.11.

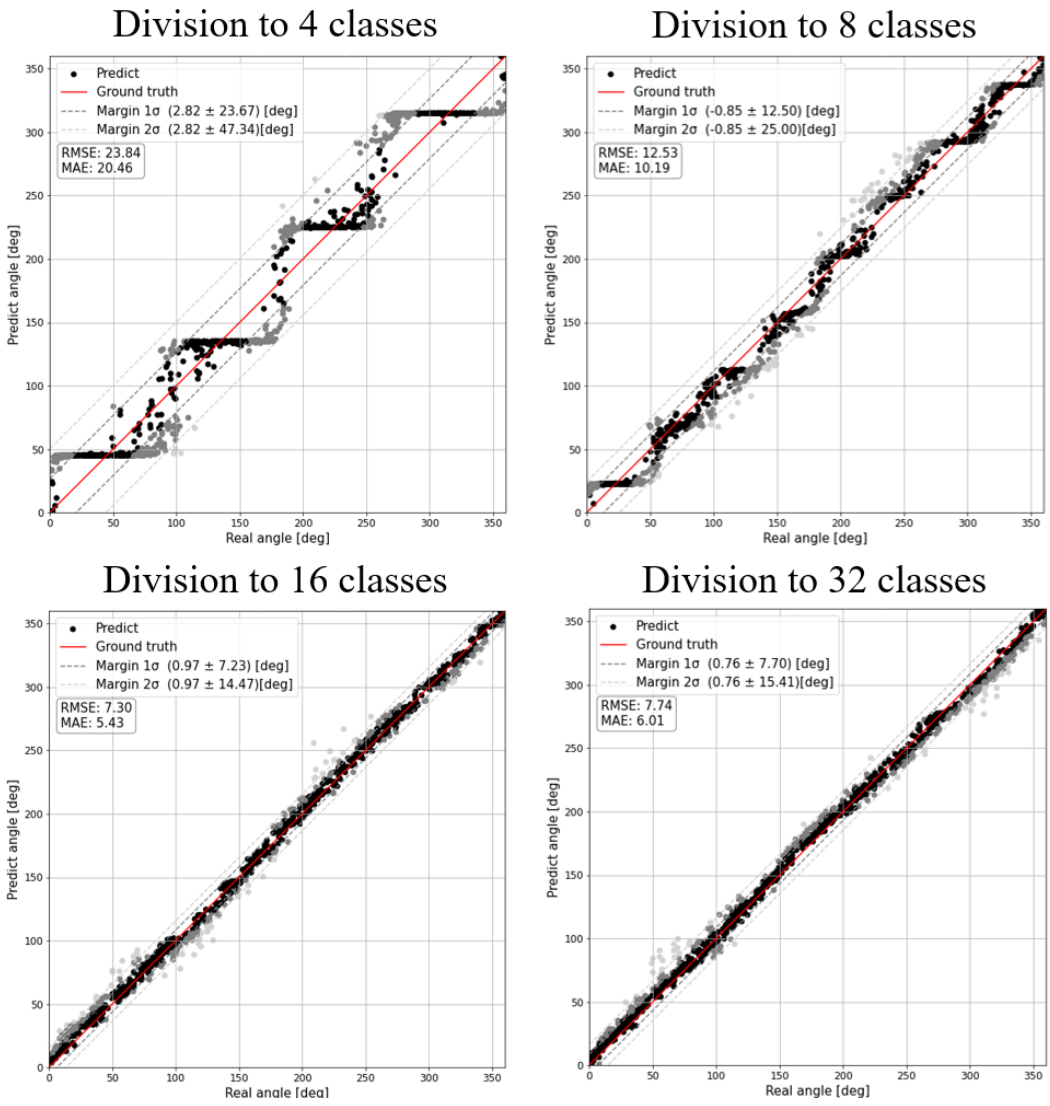


Figure 5.11: RGB angle classification results for divisions into different numbers of classes.

5.3 PIH insertion process

The insertion process is iterative as shown in Figure 5.12. In each iteration, the current 3D scene is captured, then based on the estimated relative angle and radius an insertion iteration is performed. While the estimated relative radius ($\hat{R}$) exceeds a predefined threshold, a virtual friction force ($\hat{F}$) is calculated based on the estimated relative angle ($\hat{\theta}$) (Equation 5.2) and is used in the impedance control loop to improve the peg position.

The entire process iterates until the estimated relative radius falls below the threshold. Once this condition is met, the robotic arm initiates the insertion process.

$$\hat{F} = -[sin(\hat{\theta}), cos(\hat{\theta})] \tag{5.2}$$

The threshold value is set to the maximal radial error that ensures successful insertion.

For rigid objects, the threshold value is equal to the difference between the radius of the object and the radius of the hole (Equation 5.3). On the other hand, for LDOs, insertion experiments with various radiuses were conducted to determine the threshold value.

$$Threshold = R_{hole} - R_{peg} \tag{5.3}$$

More details about the insertion process can be found at E. Cohen thesis [29].

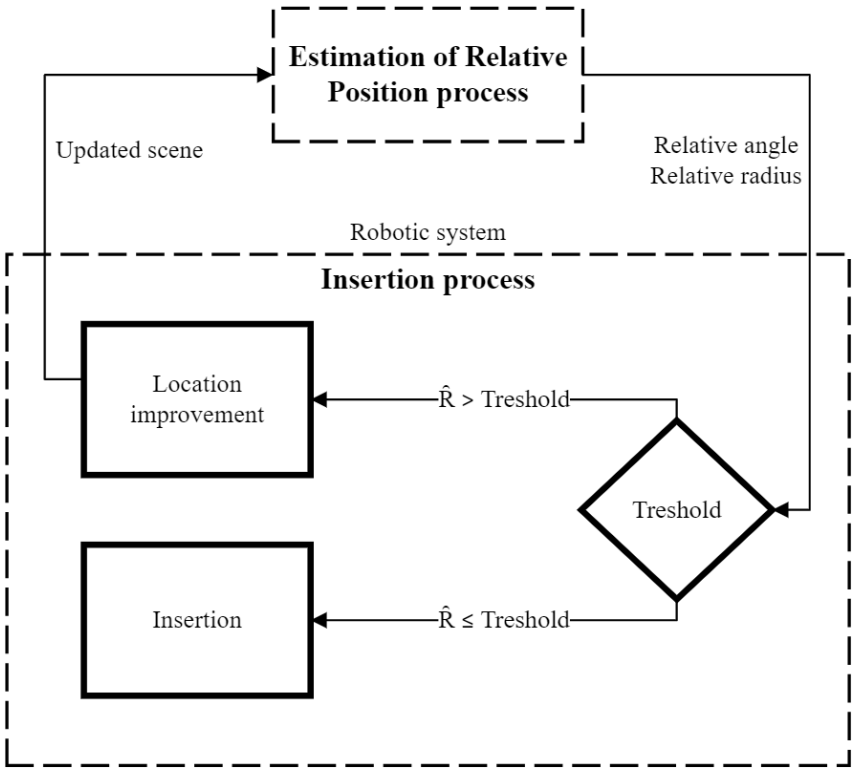


Figure 5.12: The insertion process scheme blocks.

Chapter 6

Performance analysis

In this chapter the performance of the pipeline processes outlined in the Implementation are discussed and evaluated including: (i) evaluating the training process; (ii) analyzing the estimation of relative position process; (iii) examining the impact of our approach on the insertion process.

6.1 Training process

Throughout the training process, a comprehensive performance analysis was conducted. This analysis involved comparing model performance trained on datasets captured by different cameras, examining model performance trained with and without the background filtering and discussing the tuning process of the scene division into sub-areas parameter. All the training process evaluations are based on the Medical Pipe task as it was our primary task.

6.1.1 Capturing 3D datasets

To compare the model performance trained on datasets captured by different cameras, 3D datasets for the Medical Pipe were captured with both cameras. These datasets were utilized to train the various CNN architectures. The performance of the models is summarized in Table 6.1. It appears that both cameras yield similar model performance for this task. On the other hand, when observing the datasets, it can be seen that the depth data obtained from the LiDAR camera, for example Figure 5.2, is less noisy and contains fewer holes.

Angle Estimation MAE [deg]					
Camera	**RGB**	**Depth**	**Early fusion**	**Late fusion**	**Parallel fusion**
LiDAR camera	5.43	10.86	5.37	4.75	6.96
Stereo camera	5.18	12.37	5.62	4.73	6.98

Radius Estimation MAE [mm]					
Camera	**RGB**	**Depth**	**Early fusion**	**Late fusion**	**Parallel fusion**
LiDAR camera	0.56	0.78	0.45	0.49	0.57
Stereo camera	0.47	0.77	0.46	0.49	0.50

Table 6.1: Performance of the models trained on datasets captured by both cameras for the Medical Pipe Task.

6.1.2 Data background filtering

As mentioned in the Implementation chapter, when observing the CNN activation maps it is revealed that the models focused better when the RGB-D background is flat.

Due to these findings, the background was filtered from the RGB image using the depth data. In order to examine the model performance trained with and without the background filtering, the different architectures were trained on noisy and filtered datasets. The performance evaluation is based on a test dataset that contains different backgrounds. More details about this pre-processing can be found in Appendix B. The results are presented in Table 6.2.

As expected, the best results were obtained for the models trained on the dataset with background filtering, as it is invariant to background changes.

Consequently, using background filtering contributes to generalization by enabling the network to disregard background changes in the scene.

Angle Estimation MAE [deg]					
Background	**RGB**	**Depth**	**Early fusion**	**Late fusion**	**Parallel fusion**
Noise	32.26	14.30	12.84	24.01	16.63
Filtered	10.02	16.12	5.87	5.95	10.44

Radius Estimation MAE [mm]					
Background	**RGB**	**Depth**	**Early fusion**	**Late fusion**	**Parallel fusion**
Noise	1.20	1.19	1.23	0.74	0.65
Filtered	0.50	0.95	0.76	0.45	0.60

Table 6.2: Performance of the models trained on datasets of the Medical Pipe task with and without background.

6.1.3 Scene division parameter tuning

As part of the training process all the hyper-parameters were tuned. The division parameter, which divides the scene into sub-areas, was defined as a unique hyper-parameter for our approach. Its tuning involved training the network with varying numbers of sub-areas and evaluating the estimation error. The performance is summarized in Table 6.3. It is indicated that dividing the scene into 16 sub-areas produced the best results for our dataset size.

For smaller division parameters the estimation error is high. This is according to high precision classification meaning high probability identification of the sub-areas. In such cases, the linear interpolation is not effective and the estimation is the class itself. This is also shown in Figure

Conversely, larger division parameters led to overfitting due to fewer examples per class in the training set, which resulted in reduced performance.

Angle Estimation MAE [deg]				
No' of sub areas	4	8	16	32
MAE [deg]	20.46	10.19	5.43	6.01

Table 6.3: Performance of model trained with different scene division parameters.

6.2 Estimation of relative position process

The evaluation of relative position estimation involves analyzing the performance of the various CNN architectures for each task and assessing the accuracy of the radius and angle estimation.

6.2.1 Classification CNN architectures

For each task, the classification CNN architectures were trained. These architectures were evaluated on the test set, and the results are summarized in Table 6.4 in terms of mean absolute error (MAE). The error in angle estimation is measured in degrees, and the error in radius estimation is measured in millimeters. Our approach demonstrates promising results, as both the angle and radius errors are below the predefined maximum error (1 [mm] in radius and 10° in angle). In addition, it was found that the best depth fusion method for angle estimation is late fusion, which yields a mean error of 5.3° for the classic PIH task, 6.88° for the Wiring Electric Wire task and 4.75° for the Medical Pipe task. While the radius estimation results were inexplicit, the late fusion still yielded great results.

Angle Estimation MAE [deg]					
Task	**RGB**	**Depth**	**Early fusion**	**Late fusion**	**Parallel fusion**
Classical PIH	6.65	25.38	8.43	5.30	11.78
Electric wire	25.13	22.72	11.04	6.88	12.09
Medical pipe	5.43	10.86	5.37	4.75	6.96

Radius Estimation MAE [mm]					
Task	**RGB**	**Depth**	**Early fusion**	**Late fusion**	**Parallel fusion**
Classical PIH	0.59	1.03	0.64	0.61	0.63
Electric wire	0.36	0.39	0.31	0.32	0.35
Medical pipe	0.56	0.78	0.45	0.49	0.57

Table 6.4: Relative position estimation with different architectures for the various tasks.

In addition to comparing the architectures, a comparison to Regression CNN architectures, the traditional method to estimate continuous values, was also performed. Regression CNN models were trained on the same Medical Pipe dataset used in our approach. The performance of the models is summarized in table 6.5. As can be seen, our approach estimates the angle more accurately than the regression model. Conversely, the regression model provides a better estimation of the radius. Since the estimated angle is a crucial element in the insertion process and the radius is used only as an indicator, our approach proves its effectiveness in PIH tasks.

Angle Estimation MAE [deg]					
method	**RGB**	**Depth**	**Early fusion**	**Late fusion**	**Parallel fusion**
Our approach	5.43	10.86	5.37	4.75	6.96
Regression	17.16	27.38	28.93	17.59	16.80

Radius Estimation MAE [mm]					
method	**RGB**	**Depth**	**Early fusion**	**Late fusion**	**Parallel fusion**
Our approach	0.56	0.78	0.45	0.49	0.57
Regression	0.31	1.01	0.39	0.29	0.53

Table 6.5: Performance of Regression models vs. our approach.

6.2.2 Relative position estimation

The performance of the relative position estimation was evaluated for each of the models. The performance graphs for the Medical Pipe task are presented in this chapter as it was our primary task, along with the late fusion architecture, which showed the most accurate estimation. For the other tasks and models, the results show similar behavior and presented in Appendix A.

Figure 6.1 showcases two graphs of the estimated angle (left) and estimated radius (right) against the ground truth. The red line represents the ground truth, indicating where the points should ideally lie. The gray lines indicate the first and second margins, encompassing where 95% and 68% of the points are located, respectively. The graphs show that the angle estimation yields good performance with a mean absolute error of less than 5 degrees. Additionally, while the radius estimation is less accurate, it still meets the predefined goal of less than 1 mm error.

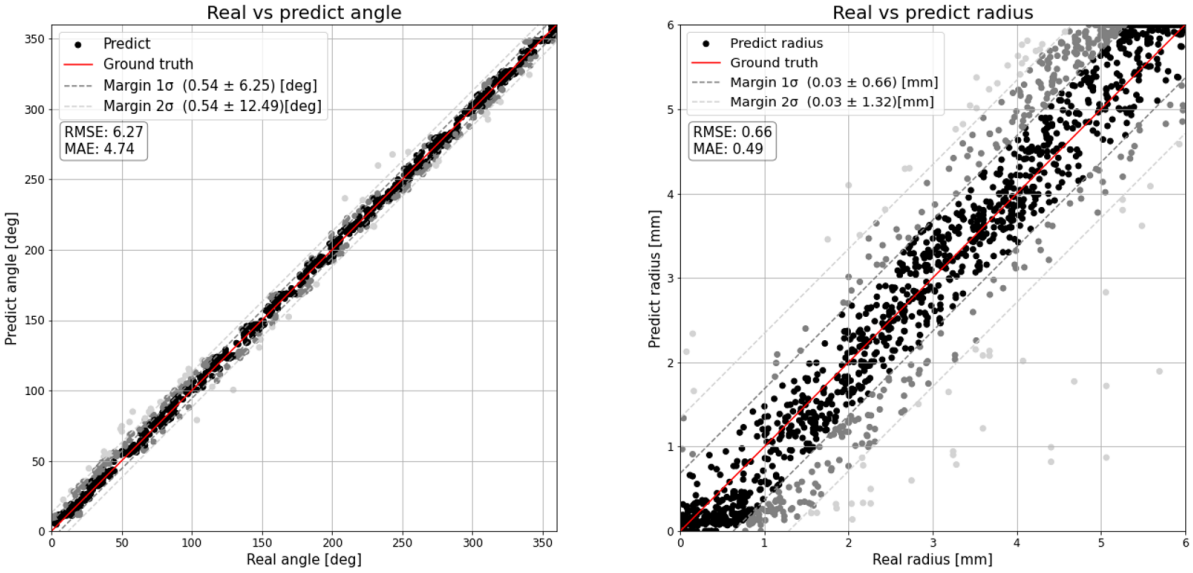


Figure 6.1: Real vs Estimated Angle (left) and Radius (right).

Since the estimated radius was used as an indicative value, it was converted into a binary value. If the radius exceeds the threshold, the binary value is classified as *Out*, indicating the object is outside the hole. Conversely, if it is below the threshold, the binary value is classified as *In*, signifying the object is inside the hole and the insertion process can be performed. The left graph in Figure 6.2 showcases the error in the estimated angle relative to the actual radius. The dark points (red and blue) represent correct predictions in terms of the in-out binary value, whereas the bright points denote incorrect predictions. The graph indicates that the larger the relative radius, the smaller the error in the estimated angle. Intuitively, as the radius decreases, determining the relative angle becomes more challenging. Additionally, the graph shows that the error in the in-out value is distributed around a 2 mm radius, reflecting the 2 mm threshold set for this task. Another perspective on this data is provided in the right graph, illustrating the prediction accuracy of the in-out values as a percentage. The red bars represent samples where the ground truth of the binary value is *In*, while the blue bars represent samples where the ground truth of the binary value is *Out*. While the red bars indicate that approximately 5%

of the samples were incorrectly predicted, these results do not impact the process. The arm will continue to adjust its position until an *In* prediction is received. Conversely, the blue bars show that approximately 8% of the samples were incorrectly predicted. In these cases, the robotic arm will initiate the insertion process outside the hole, leading to a failed insertion. To minimize this error, the insertion process is initiated only after receiving two *In* consecutive predictions, reducing the chance of inserting the object outside the hole to below 1%.

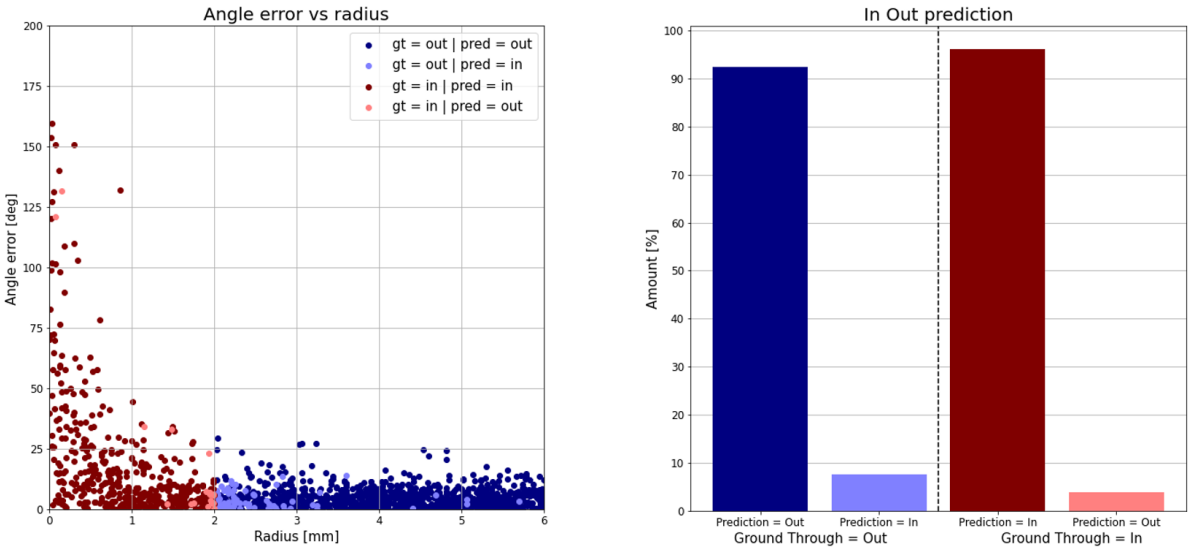


Figure 6.2: Angle error vs Radius (left) and In Out prediction (right).

AAnother way to represent the estimation error in the radius and angle is in the Cartesian coordinate system. The graphs in Figure 6.3 showcase the estimation error in the angle (left) and the radius (right) based on the peg location on the insertion plane, where the origin represents the center of the hole, and the points indicate the center of the peg. The radius error graph (right) shows that the error in the radius is not correspond to the peg location. The error may arise from factors like image quality and capturing angle. Additionally, as previously shown, the estimated angle error (left) is larger for small radii. However, in our case, the insertion process is already initiated in these radii. Examining the rest of the angle errors, it can be seen that it distributed similarly to the radius estimation error as poor image quality would likely affect both parameters.

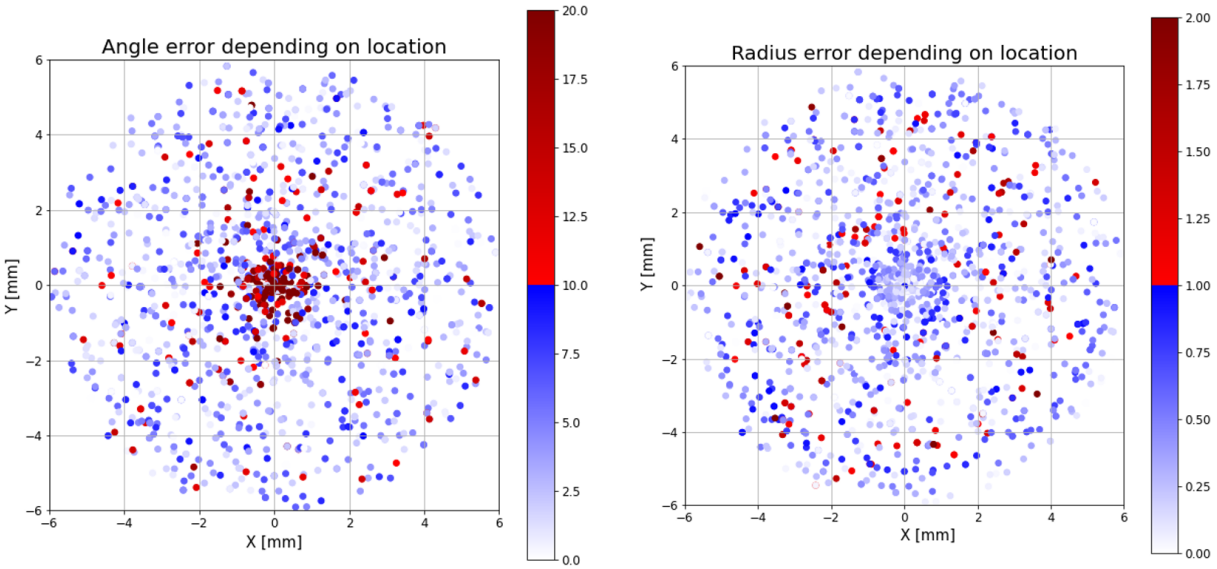


Figure 6.3: Angle error (left) and Radius error (right) depending on location.

To validate the real-world applicability and the generalization of the method, a performance test was conducted in an industrial setting. This test focused on the Medical Pipe task and was executed without additional training. The experiments took place at "Polygon Technologies" which collaborated with our lab as part of the ART (Assembly by Robotic Technology) program - the Israel Innovations Authority. The results of these experiments are depicted in the graphs in Figure 6.4. As evident, the results were promising even without additional training, highlighting the generalization capability of our method.

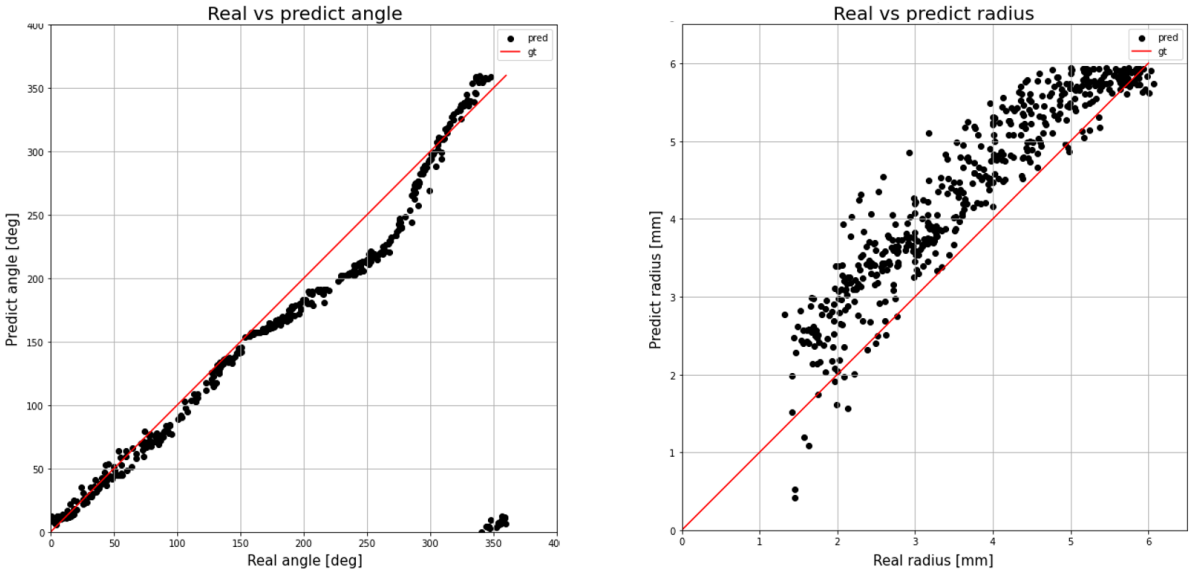


Figure 6.4: Real vs Estimated Angle (left) and Radius (right) in the industry.

6.3 PIH insertion process

The true test of our method lies in validating it across the entire insertion pipeline. For this validation, Eylon Cohen conducted insertion experiments both with and without utilizing our relative position estimation approach. The results are summarized in Table 6.6.

As can be seen in the table, for rigid PIH task with a 4.5 mm peg and a 6 mm hole, our method enhanced the success rate from 83% to 100%. Similar improvements were observed for an 8 mm peg and a 10 mm hole. For a 16 mm peg and a 20 mm hole, the success rate rose from 51% to 90%. Additionally, for the Medical Pipe task, the success rate in the insertion process increased from 80% to 98%.

Furthermore, the experiments with our method were conducted with increased initial radial uncertainty in the hole position. For example, in PIH task (8mm peg / 10 mm hole) the radial uncertainty without using our method was set to maximum of 3.5 mm while with our method it was set to maximum of 10 mm, and still rase the success from 80% to 100%. Further validating the effectiveness of our approach.

Insertion success rate [%]			
Task	**With/Without our method**	**Radial uncertainty in hole position [mm]**	**Success rate [%] (Out of 100 trials)**
4.5[mm] Peg 6[mm] Hole	Without	[1, 2.5]	83
	With	[1, 3]	100
8[mm] Peg 10[mm] Hole	Without	[1.5, 3.5]	80
	With	[1.5, 10]	100
16[mm] Peg 20[mm] Hole	Without	[2.5, 8]	51
	With	[4, 14]	90
8.5[mm] Medical pipe 8.5[mm] Connector	Without	[1.5, 2.5]	80
	With	[1.5, 2.5]	95

Table 6.6: Success rate of the insertion process with and without visual sensor.

Chapter 7

Conclusion and further work

This research presents a classification-based approach for estimating the relative position between objects, focusing on Linear Deformable Objects (LDOs). The objective was to develop and evaluate the solution for various Peg-in-Hole (PIH) assembly tasks in a real industrial robotic environment. The approach included capturing RGB-D data, estimating relative positions by discretizing the scene space into classes, classifying the probability vector of each class, and estimating continuous relative positions using linear interpolation. Insertion operations were then performed based on the estimated relative positions. The approach underwent testing in an industrial setting at Polygon Technologies as part of the ART (Assembly by Robotic Technology) project managed by the Israel Innovation Authority. These tests yielded promising results, demonstrating the method's effectiveness even without additional training. In addition to evaluating the approach across various Peg-in-Hole (PIH) tasks, the research involved implementing and evaluating different depth data fusion CNN architectures. Furthermore, a comparison was made between two 3D capturing technologies. Following are the insights and conclusions from the research:

Data capturing: Overall the data captured with both cameras was of good quality and yielded great results. With respect to the tasks in this research's scope, the depth data captured by the LiDAR camera was less noisy and contained fewer gaps compared to the stereo camera. In general, LiDAR cameras struggle with capturing depth for shiny or transparent objects, while stereo cameras face challenges with texture-less or small-scale objects, such as the objects in the wiring 1[mm] wire task. However, for tasks that both cameras were able to perform, they yielded similar results. Therefore, the optimal choice between LiDAR and stereo cameras depends on the nature of the task.

RGB-D data fusion: The comparison of depth fusion methods revealed that, for angle estimation, the most efficient fusion is the late fusion method where the fusion is done within the CNN. Furthermore, it showed that naive fusion methods like parallel fusion can reduce the accuracy, even compared to networks utilizing only RGB images. In future, it would be recommended to explore additional fusion methods for integrating depth data within the CNN architecture.

Estimate the relative position: For all tasks, satisfactory results were achieved in both angle and radius estimation. Our approach yielded high precision in the estimation of the relative angle compared to conventional methods. However, for the radius estimation, the regression-based method outperformed our approach. It can be attributed to the limitations of our linear interpolation implementation where the relative angles are a circular domain while the radius domain has bounds that cannot be fully captured. Although radius estimation was primarily used for the insertion criterion in our PIH tasks, it may play a more crucial role in other scenarios. Therefore, future work could explore alternative interpolation techniques or integrate continuous radius prediction directly into the classification network architecture.

PIH insertion tasks: One of the most important evaluations of our approach was the insertion experiments. Those experiments tested the applicability of our approach in the real-world industry environment. This evaluation involved the insertion of rigid and non-rigid PIH. The outcomes demonstrated a significant improvement in the success rate of the insertion operation across all tasks.

Chapter 8

References

[1] M. T. Shahria, M. S. H. Sunny, M. I. I. Zarif, J. Ghommam, S. I. Ahamed, and M. H. Rahman, "A comprehensive review of vision-based robotic applications: Current state, components, approaches, barriers, and potential solutions," *Robotics*, vol. 11, no. 6, p. 139, 2022.

[2] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[3] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

[4] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 1–10.

[5] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[10] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li, "Development of convolutional neural network and its application in image classification: A survey," *Optical Engineering*, vol. 58, no. 4, pp. 040 901–040 901, 2019.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud, "A comprehensive analysis of deep regression," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 9, pp. 2065–2081, 2019.

[13] A. X. Chang, T. Funkhouser, L. Guibas, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[15] M. Gao, J. Jiang, G. Zou, V. John, and Z. Liu, "Rgb-d-based object recognition using multimodal convolutional neural networks: A survey," *IEEE access*, vol. 7, pp. 43 110–43 136, 2019.

[16] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 681–687.

[17] L. A. Alexandre, "3d object recognition using convolutional neural networks with transfer learning between input channels," in *Intelligent Autonomous Systems 13: Proceedings of the 13th International Conference IAS-13*, Springer, 2016, pp. 889–898.

[18] J. Jiang, Z. Huang, Z. Bi, X. Ma, and G. Yu, "State-of-the-art control strategies for robotic pih assembly," *Robotics and Computer-Integrated Manufacturing*, vol. 65, p. 101 894, 2020.

[19] X. Yanchun, B. Yuewei, and H. Yafei, "Assembly strategy study on the elastic deformable peg in hole," in *2010 The 2nd International Conference on Industrial Mechatronics and Automation*, IEEE, vol. 1, 2010, pp. 193–197.

[20] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, and A. M. Agogino, "Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 2062–2069.

[21] N. Hogan, "Impedance control: An approach to manipulation: Part ii—implementation," 1985.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[23] Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba, "A review on reinforcement learning for contact-rich robotic manipulation tasks," *Robotics and Computer-Integrated Manufacturing*, vol. 81, p. 102 517, 2023.

[24] H. F. Al-Shuka, S. Leonhardt, W.-H. Zhu, R. Song, C. Ding, Y. Li, *et al.*, "Active impedance control of bioinspired motion robotic manipulators: An overview," *Applied bionics and biomechanics*, vol. 2018, 2018.

[25] L. Bodenhagen, A. R. Fugl, M. Willatzen, H. G. Petersen, and N. Krüger, "Learning peg-in-hole actions with flexible objects.," in *ICAART (1)*, 2012, pp. 624–631.

[26] L. Bodenhagen, A. R. Fugl, A. Jordt, *et al.*, "An adaptable robot vision system performing manipulation actions with flexible objects," *IEEE transactions on automation science and engineering*, vol. 11, no. 3, pp. 749–765, 2014.

[27] H.-C. Song, Y.-L. Kim, and J.-B. Song, "Automated guidance of peg-in-hole assembly tasks for complex-shaped parts," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2014, pp. 4517–4522.

[28] Y. Shen, Q. Jia, R. Wang, Z. Huang, and G. Chen, "Learning-based visual servoing for high-precision peg-in-hole assembly," in *Actuators*, MDPI, vol. 12, 2023, p. 144.

[29] E. Cohen, "Learning robotic assembly policies with non-co-aligned force/torque sensor and an optional camera," Master's thesis, Technion - Israel Institute of Technology, 2023.

Appendix A

Performance analysis graphs of all tasks and models

The performance of the relative position estimation was evaluated for each of the models. In Chapter 6, performance graphs for the Medical Pipe task, along with the late fusion architecture, are presented.

In this appendix, performance graphs for all the models, including the various tasks: Classical PIH, Electric wire and Medical Pipe tasks and fusion architectures: RGB, Depth, Early fusion, Late fusion and Parallel fusion, are presented.

Classical PIH task - RGB architecture:

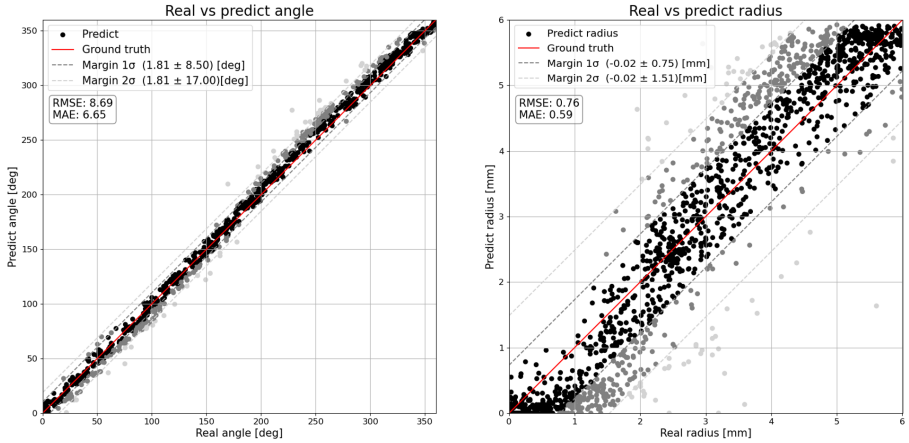


Figure A.1: Real vs Estimated Angle (left) and Radius (right) for the Classical PIH task and the RGB architecture.

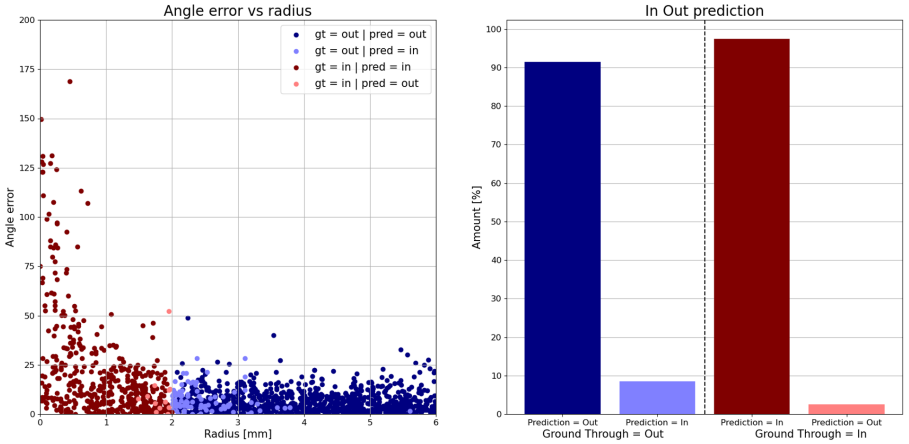


Figure A.2: Angle error vs Radius (left) and In Out prediction (right) for the Classical PIH task and the RGB architecture.

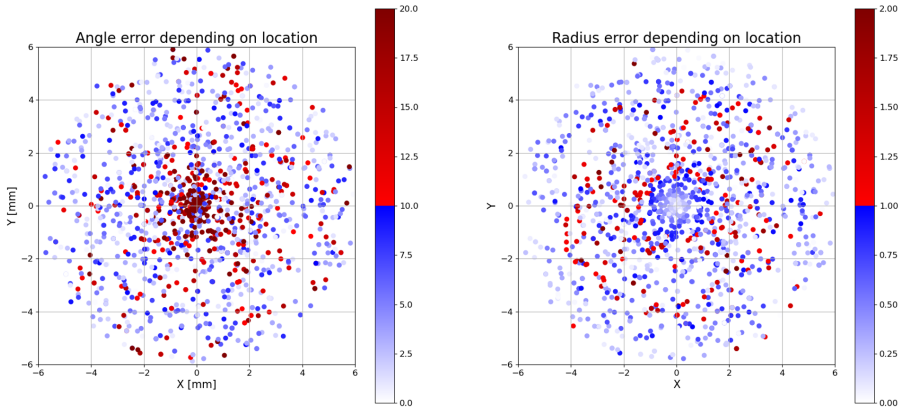


Figure A.3: Angle error (left) and Radius error (right) depending on location for the Classical PIH task and the RGB architecture.

Classical PIH task - Depth architecture:

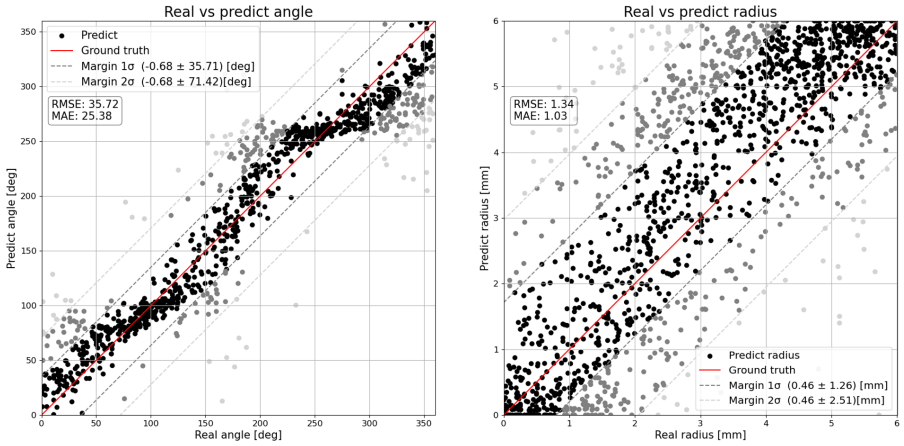


Figure A.4: Real vs Estimated Angle (left) and Radius (right) for the Classical PIH task and the Depth architecture.

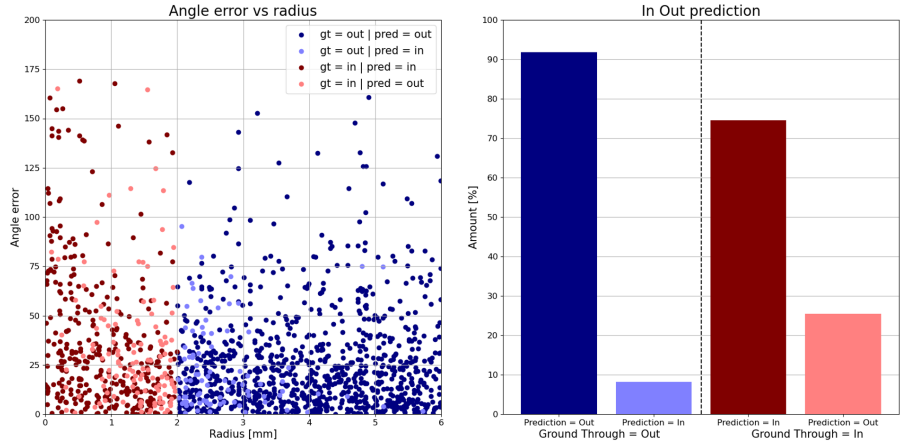


Figure A.5: Angle error vs Radius (left) and In Out prediction (right) for the Classical PIH task and the Depth architecture.

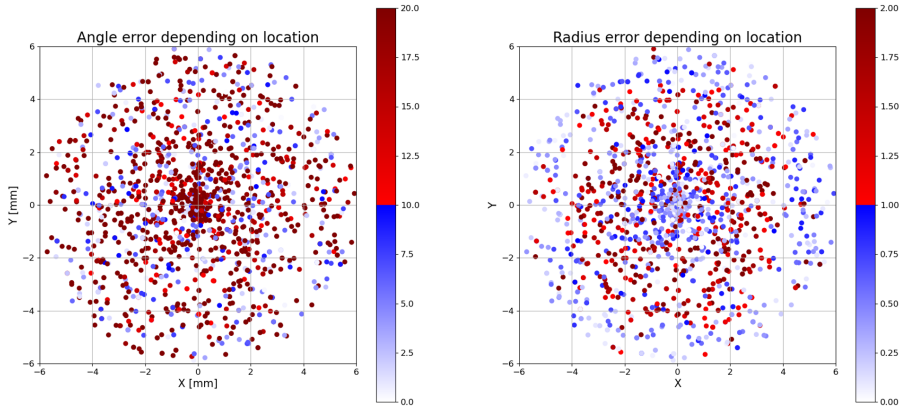


Figure A.6: Angle error (left) and Radius error (right) depending on location for the Classical PIH task and the Depth architecture.

Classical PIH task - Early fusion architecture:

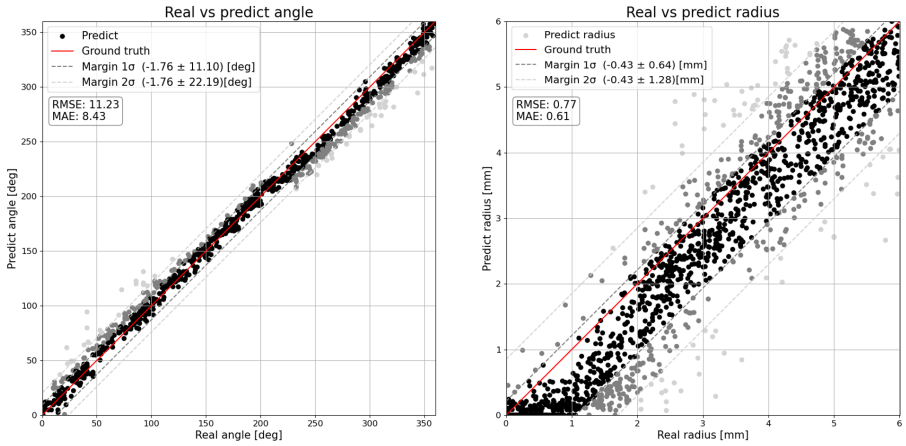


Figure A.7: Real vs Estimated Angle (left) and Radius (right) for the Classical PIH task and the Early fusion architecture.

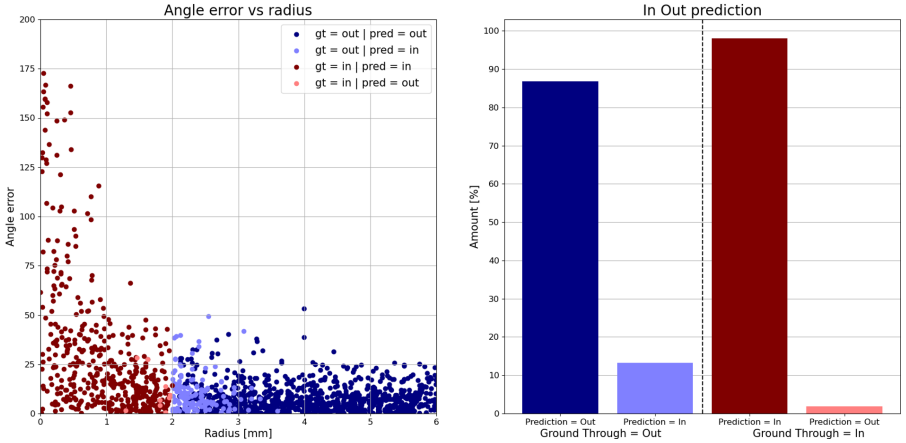


Figure A.8: Angle error vs Radius (left) and In Out prediction (right) for the Classical PIH task and the Early fusion architecture.

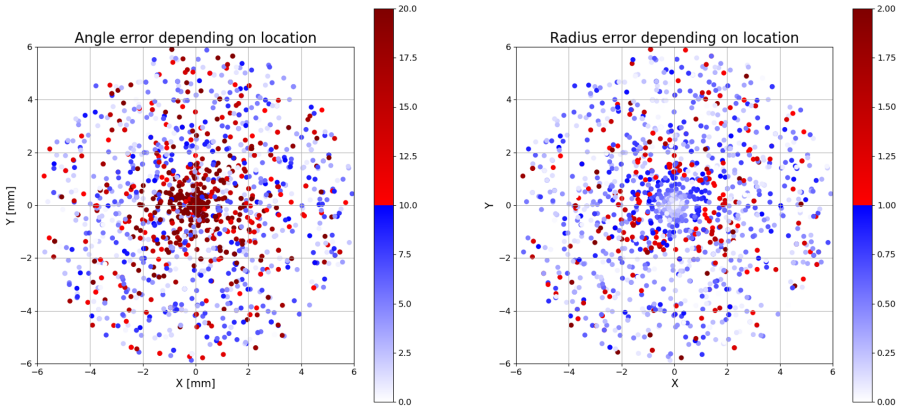


Figure A.9: Angle error (left) and Radius error (right) depending on location for the Classical PIH task and the Early fusion architecture.

Classical PIH task - Late fusion architecture:

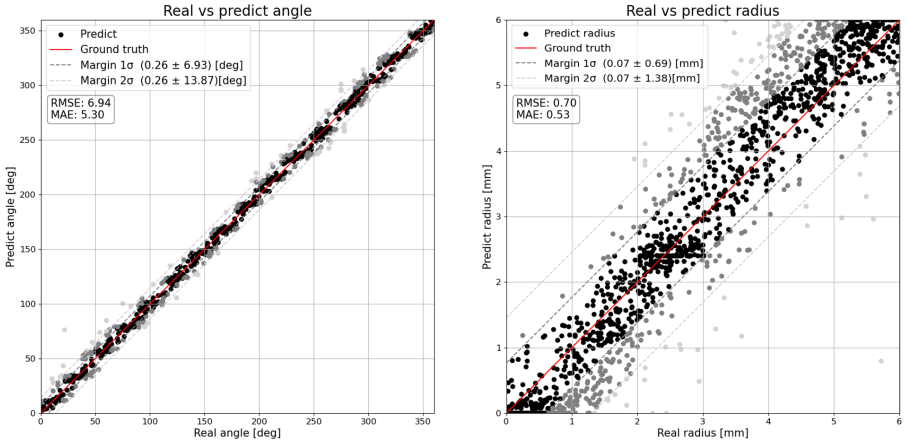


Figure A.10: Real vs Estimated Angle (left) and Radius (right) for the Classical PIH task and the Late fusion architecture.

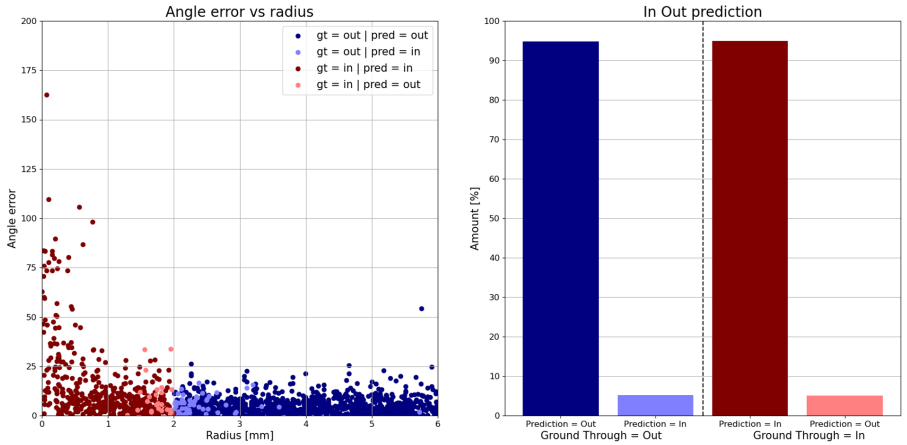


Figure A.11: Angle error vs Radius (left) and In Out prediction (right) for the Classical PIH task and the Late fusion architecture.

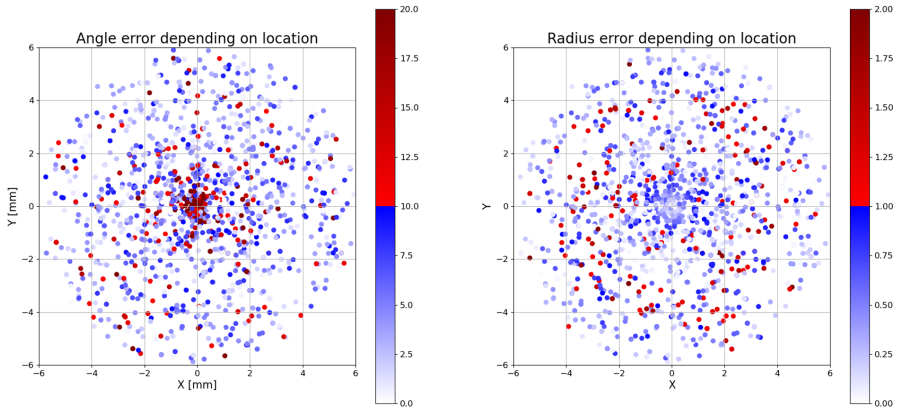


Figure A.12: Angle error (left) and Radius error (right) depending on location for the Classical PIH task and the Late fusion architecture.

Classical PIH task - Parallel fusion architecture:

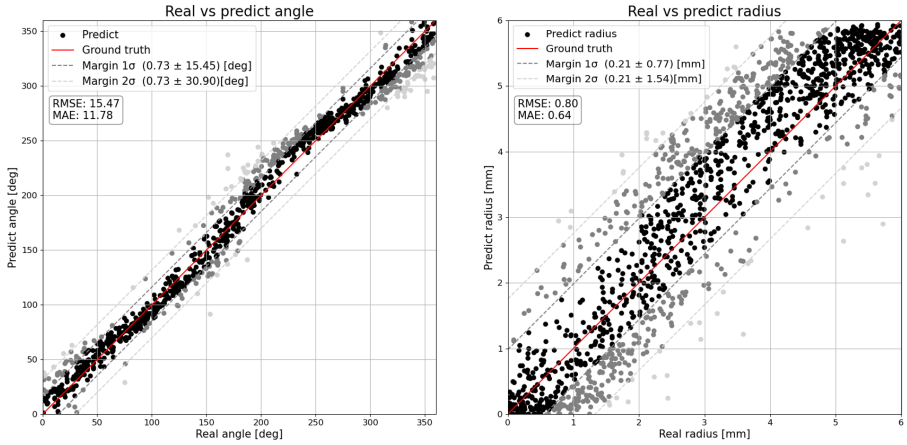


Figure A.13: Real vs Estimated Angle (left) and Radius (right) for the Classical PIH task and the Parallel fusion architecture.

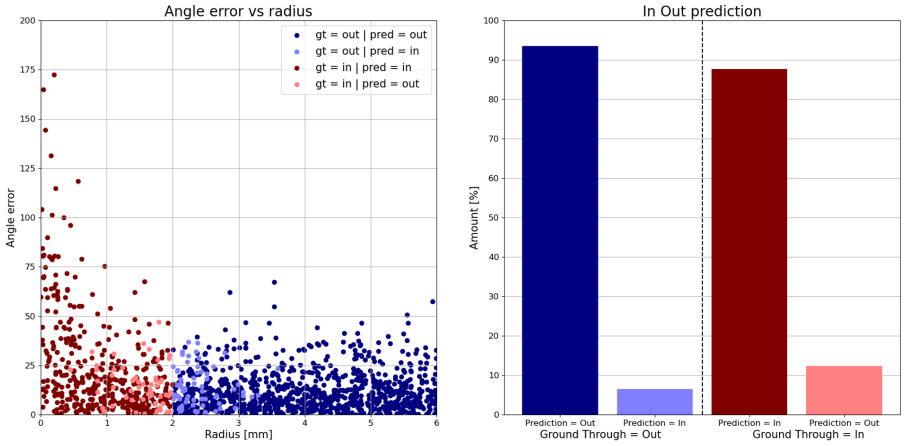


Figure A.14: Angle error vs Radius (left) and In Out prediction (right) for the Classical PIH task and the Parallel fusion architecture.

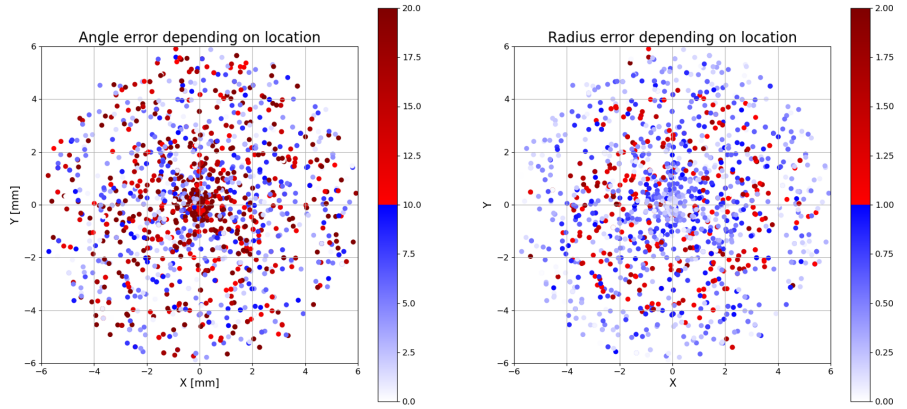


Figure A.15: Angle error (left) and Radius error (right) depending on location for the Classical PIH task and the Parallel fusion architecture.

Electric wire - RGB architecture:

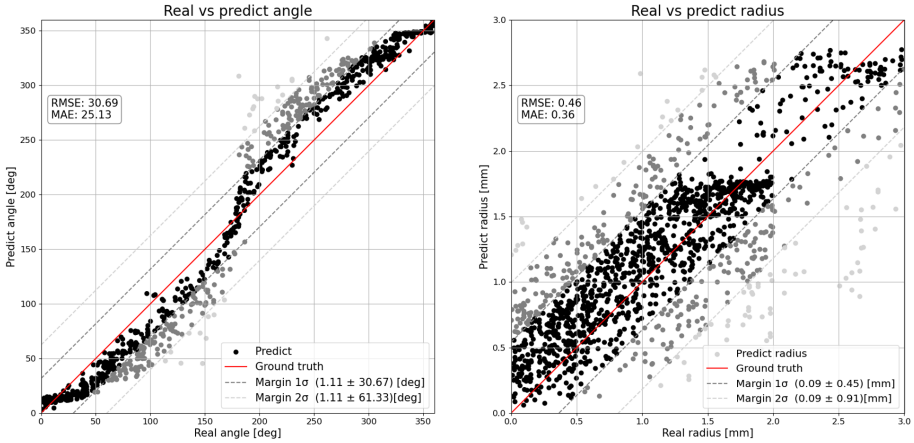


Figure A.16: Real vs Estimated Angle (left) and Radius (right) for the Electric wire and the RGB architecture.

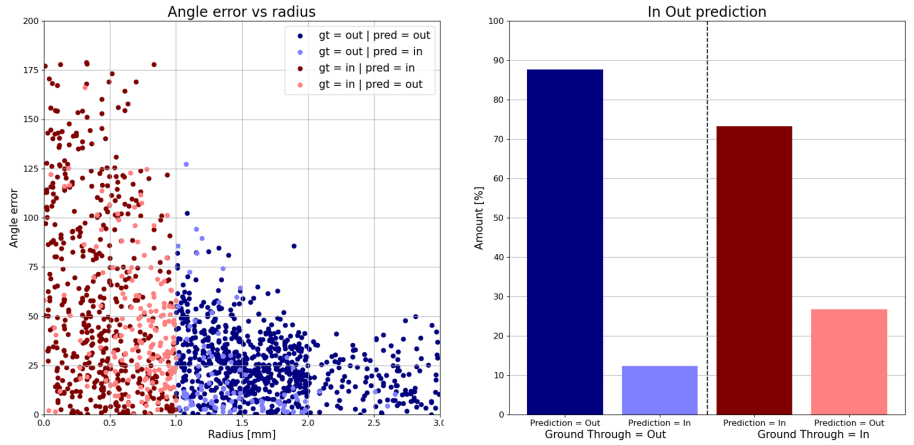


Figure A.17: Angle error vs Radius (left) and In Out prediction (right) for the Electric wire and the RGB architecture.

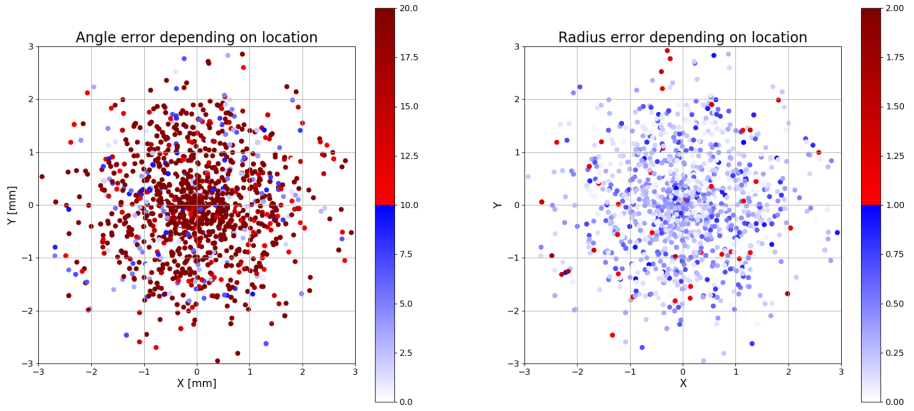


Figure A.18: Angle error (left) and Radius error (right) depending on location for the Electric wire and the RGB architecture.

Electric wire - Depth architecture:

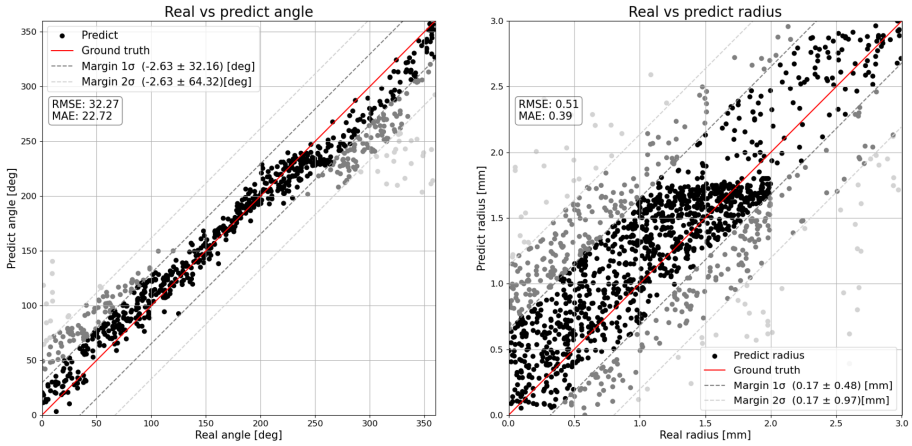


Figure A.19: Real vs Estimated Angle (left) and Radius (right) for the Electric wire and the Depth architecture.

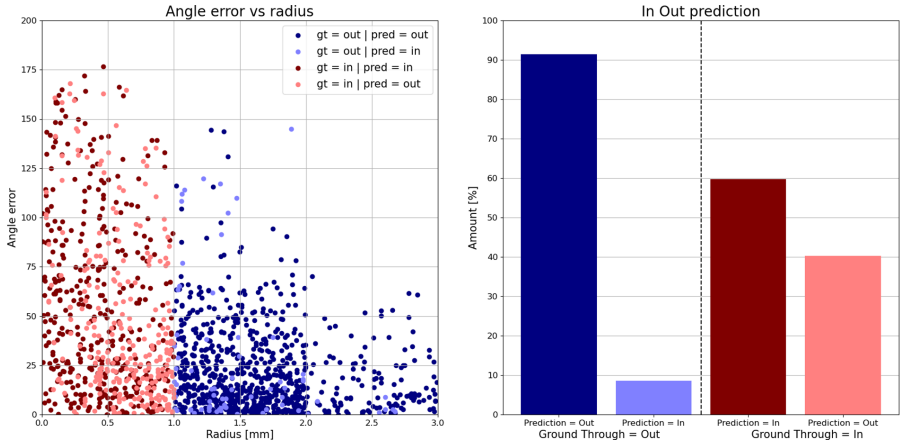


Figure A.20: Angle error vs Radius (left) and In Out prediction (right) for the Electric wire and the Depth architecture.

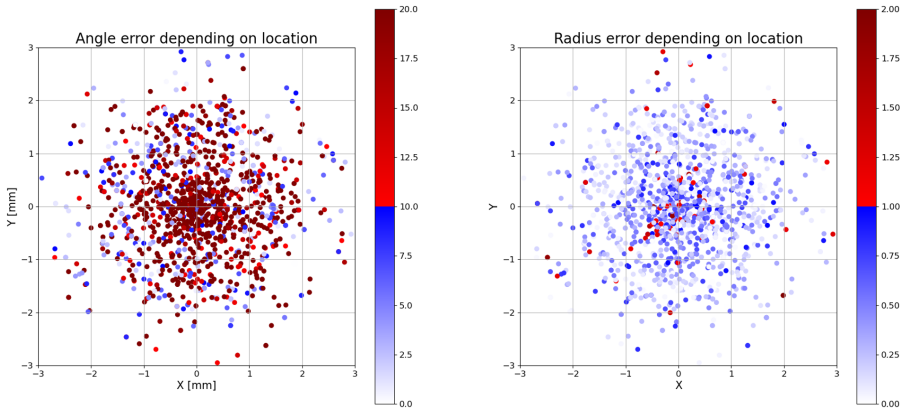


Figure A.21: Angle error (left) and Radius error (right) depending on location for the Electric wire and the Depth architecture.

Electric wire - Early fusion architecture:

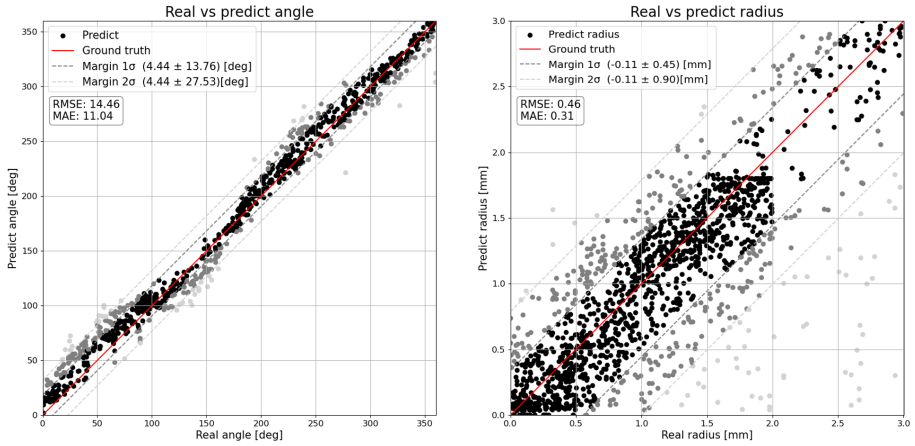


Figure A.22: Real vs Estimated Angle (left) and Radius (right) for the Electric wire and the Early fusion architecture.

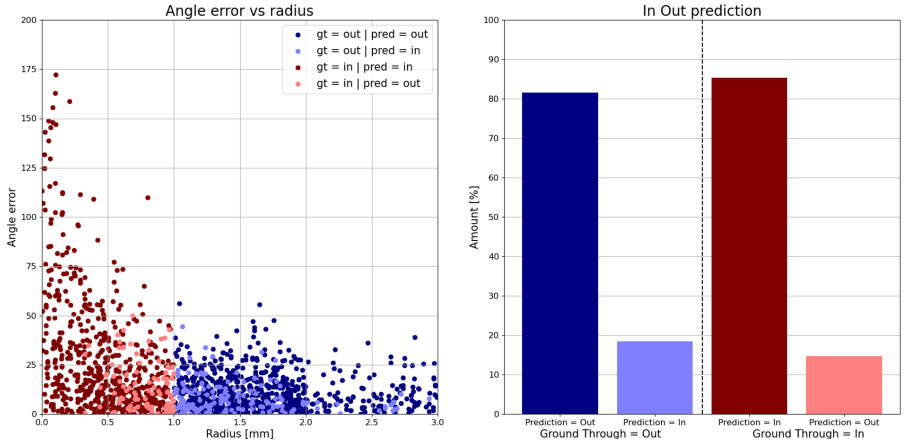


Figure A.23: Angle error vs Radius (left) and In Out prediction (right) for the Electric wire and the Early fusion architecture.

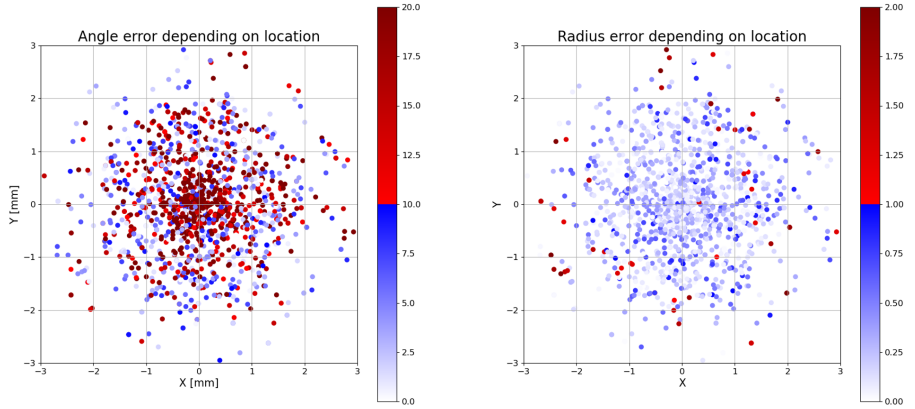


Figure A.24: Angle error (left) and Radius error (right) depending on location for the Electric wire and the Early fusion architecture.

Electric wire - Late fusion architecture:

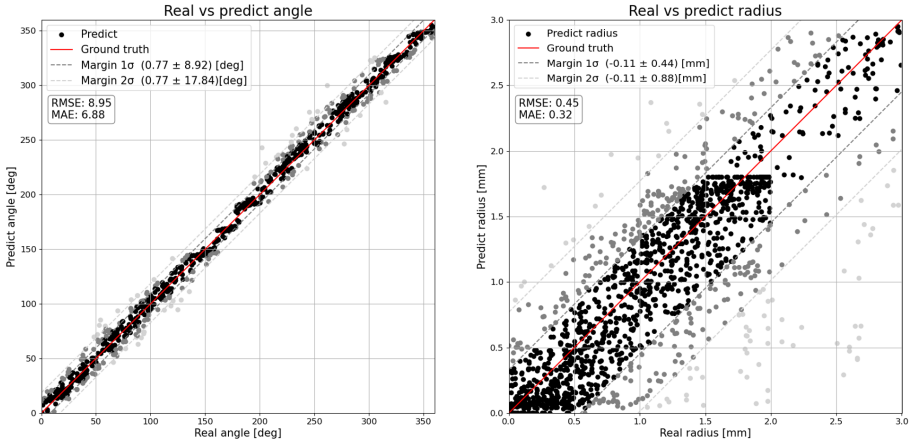


Figure A.25: Real vs Estimated Angle (left) and Radius (right) for the Electric wire and the Late fusion architecture.

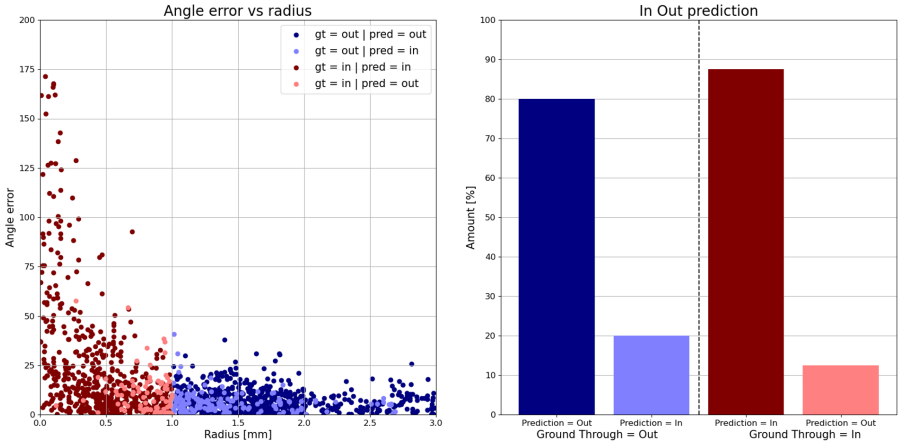


Figure A.26: Angle error vs Radius (left) and In Out prediction (right) for the Electric wire and the Late fusion architecture.

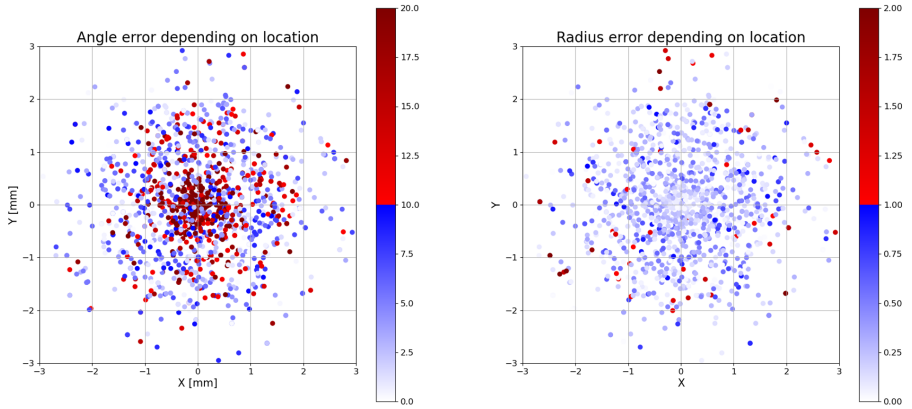


Figure A.27: Angle error (left) and Radius error (right) depending on location for the Electric wire and the Late fusion architecture.

Electric wire - Parallel fusion architecture:

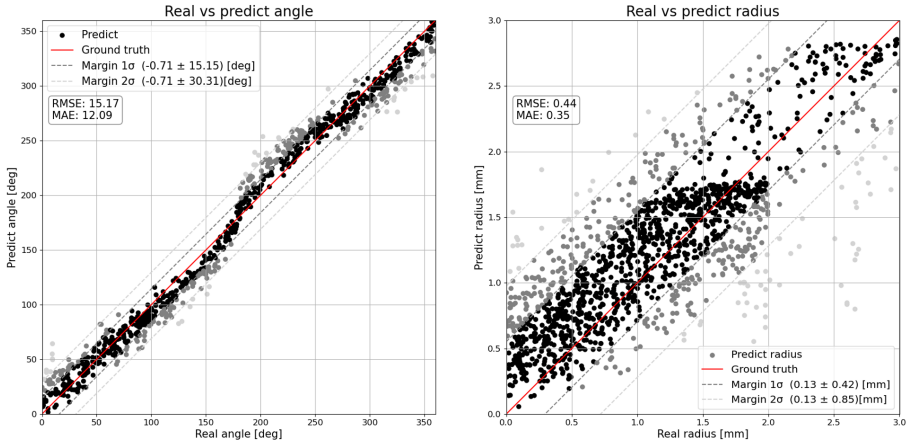


Figure A.28: Real vs Estimated Angle (left) and Radius (right) for the Electric wire and the Parallel fusion architecture.

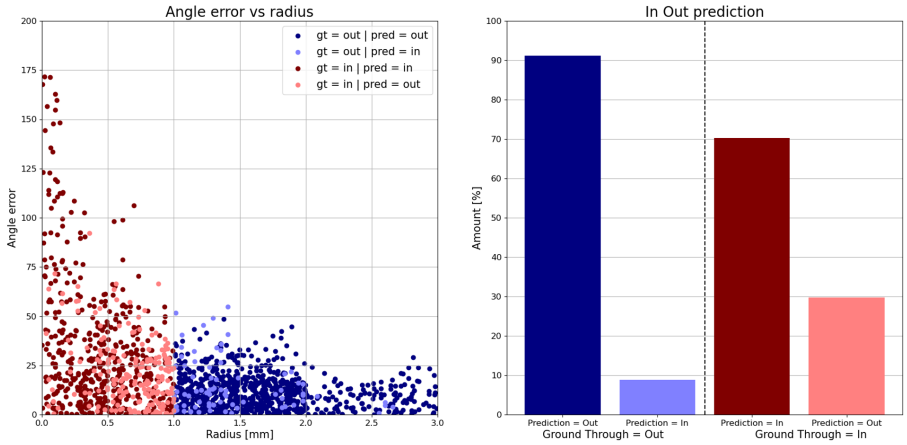


Figure A.29: Angle error vs Radius (left) and In Out prediction (right) for the Electric wire and the Parallel fusion architecture.

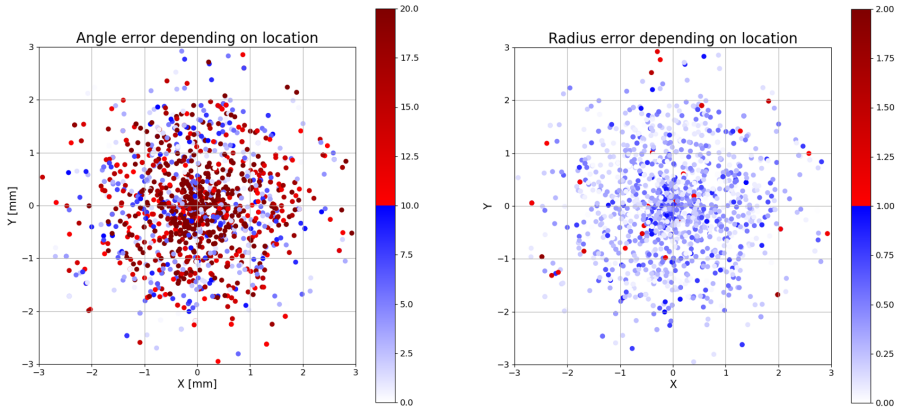


Figure A.30: Angle error (left) and Radius error (right) depending on location for the Electric wire and the Parallel fusion architecture.

Medical Pipe task - RGB architecture:

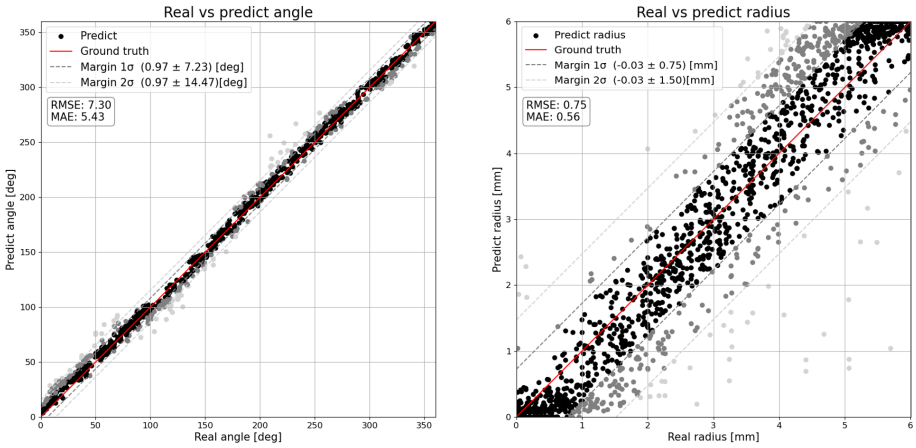


Figure A.31: Real vs Estimated Angle (left) and Radius (right) for the Medical Pipe task and the RGB architecture.

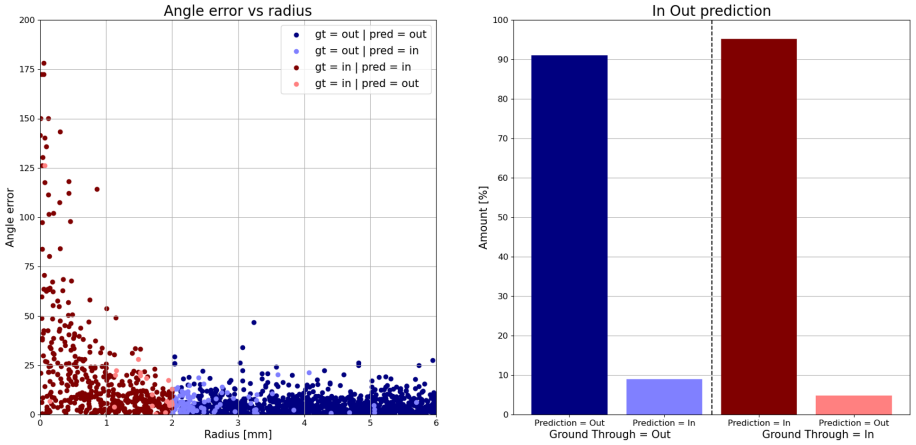


Figure A.32: Angle error vs Radius (left) and In Out prediction (right) for the Medical Pipe task and the RGB architecture.

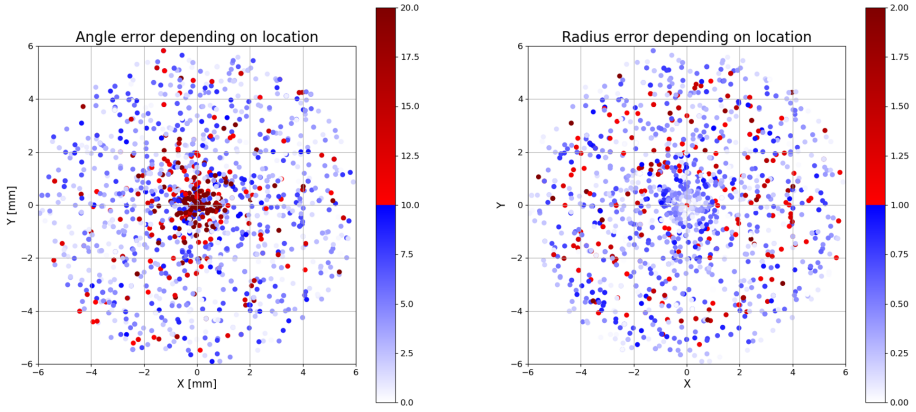


Figure A.33: Angle error (left) and Radius error (right) depending on location for the Medical Pipe task and the RGB architecture.

Medical Pipe task - Depth architecture:

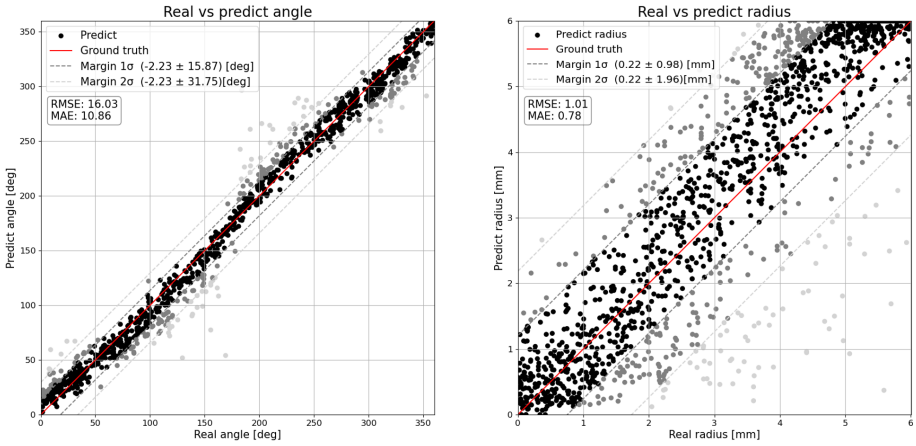


Figure A.34: Real vs Estimated Angle (left) and Radius (right) for the Medical Pipe task and the Depth architecture.

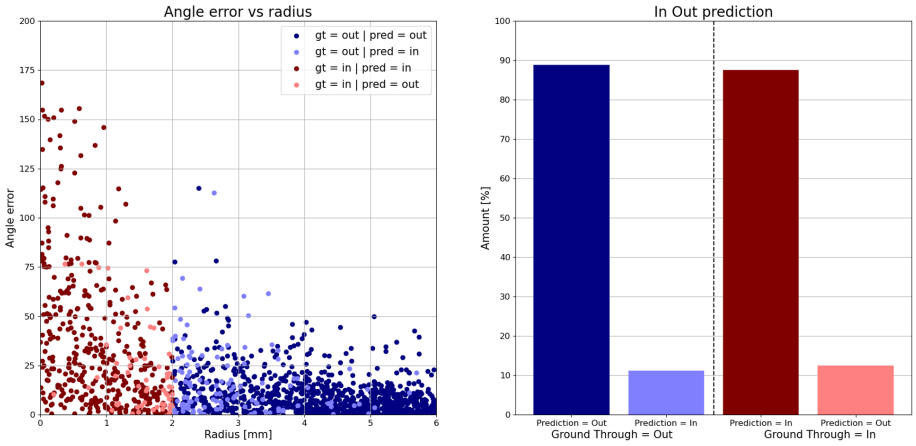


Figure A.35: Angle error vs Radius (left) and In Out prediction (right) for the Medical Pipe task and the Depth architecture.

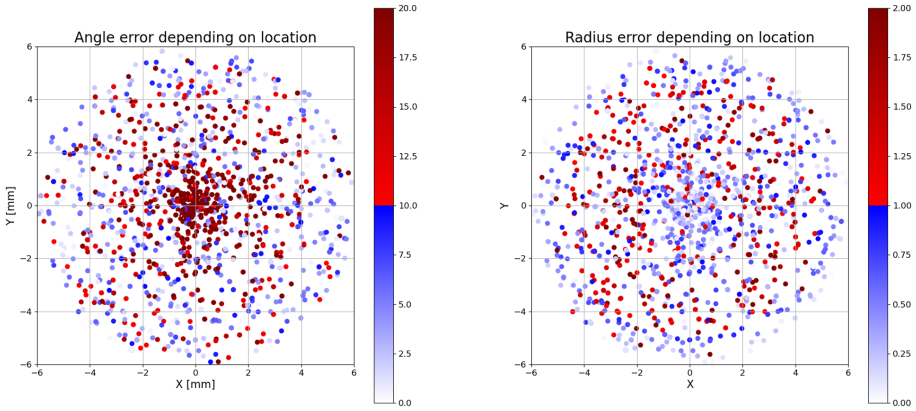


Figure A.36: Angle error (left) and Radius error (right) depending on location for the Medical Pipe task and the Depth architecture.

Medical Pipe task - Early fusion architecture:

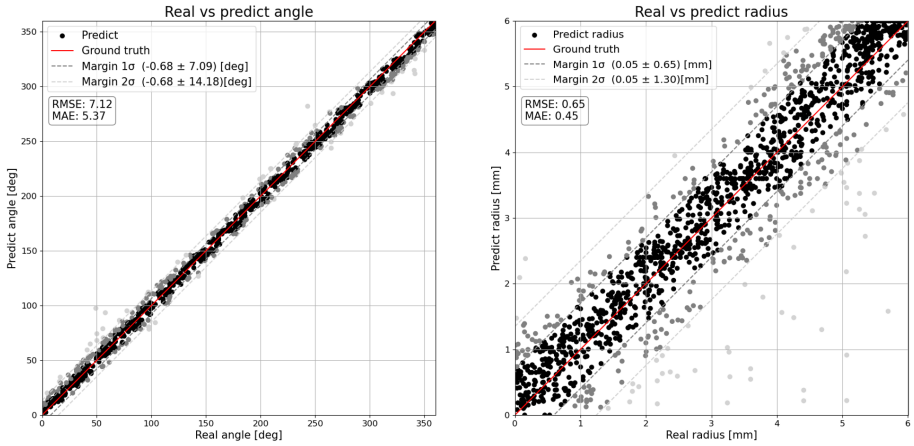


Figure A.37: Real vs Estimated Angle (left) and Radius (right) for the Medical Pipe task and the Early fusion architecture.

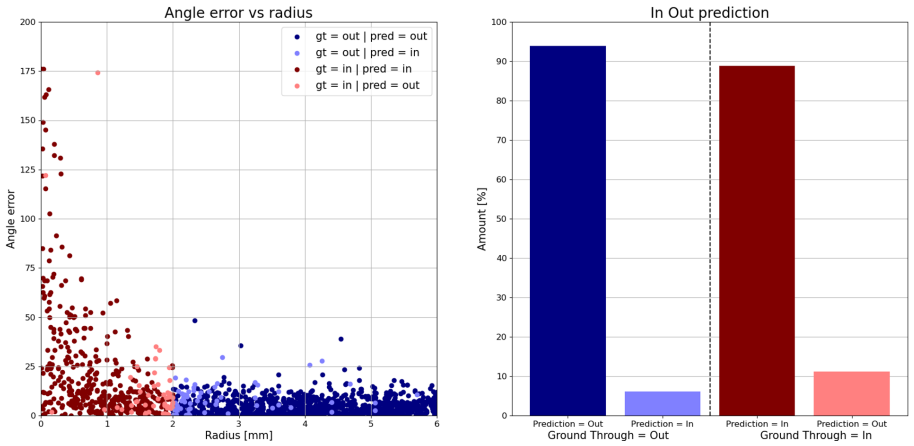


Figure A.38: Angle error vs Radius (left) and In Out prediction (right) for the Medical Pipe task and the Early fusion architecture.

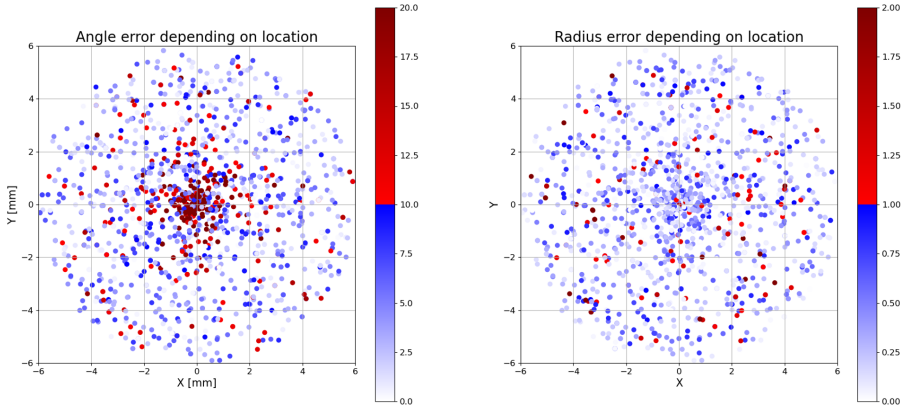


Figure A.39: Angle error (left) and Radius error (right) depending on location for the Medical Pipe task and the Early fusion architecture.

Medical Pipe task - Late fusion architecture:

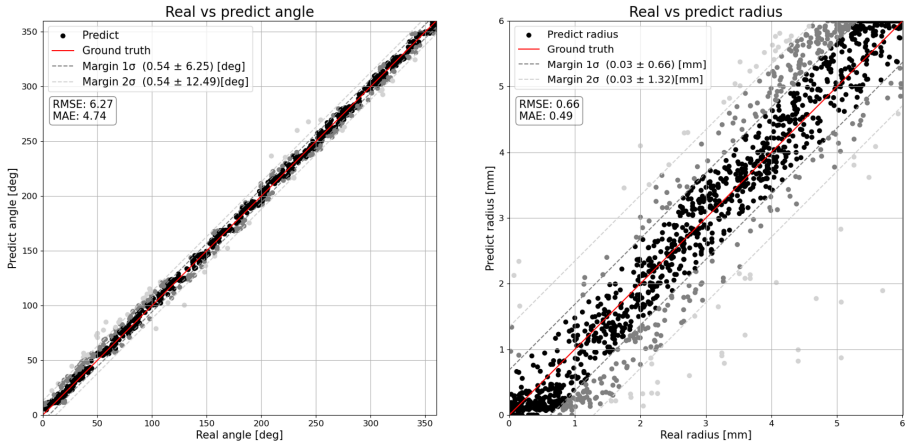


Figure A.40: Real vs Estimated Angle (left) and Radius (right) for the Medical Pipe task and the Late fusion architecture.

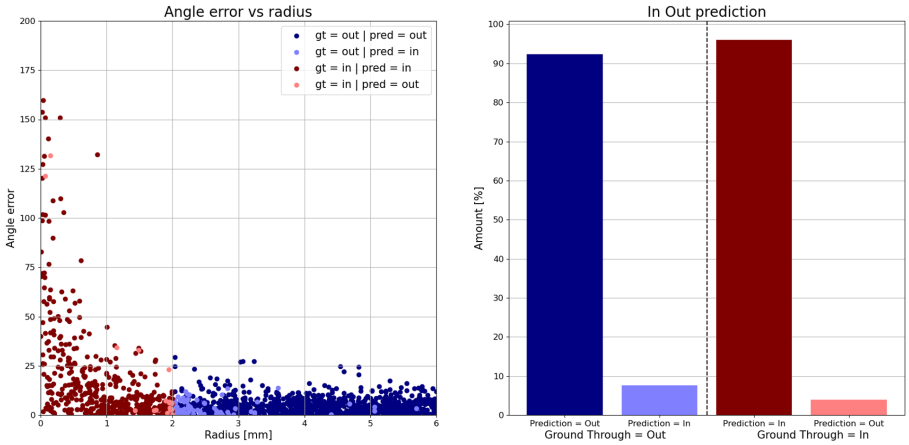


Figure A.41: Angle error vs Radius (left) and In Out prediction (right) for the Medical Pipe task and the Late fusion architecture.

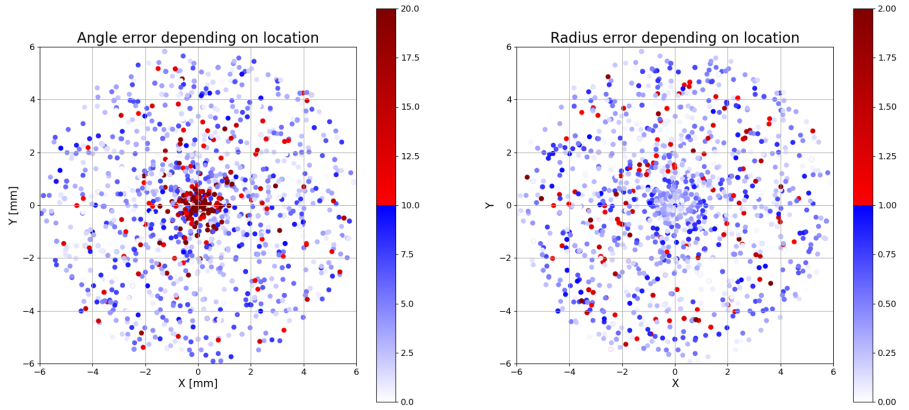


Figure A.42: Angle error (left) and Radius error (right) depending on location for the Medical Pipe task and the Late fusion architecture.

Medical Pipe task - Parallel fusion architecture:

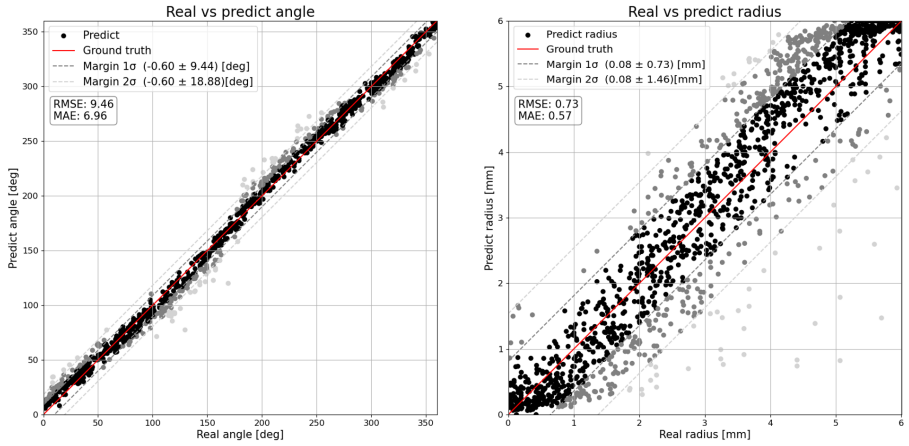


Figure A.43: Real vs Estimated Angle (left) and Radius (right) for the Medical Pipe task and the Parallel fusion architecture.

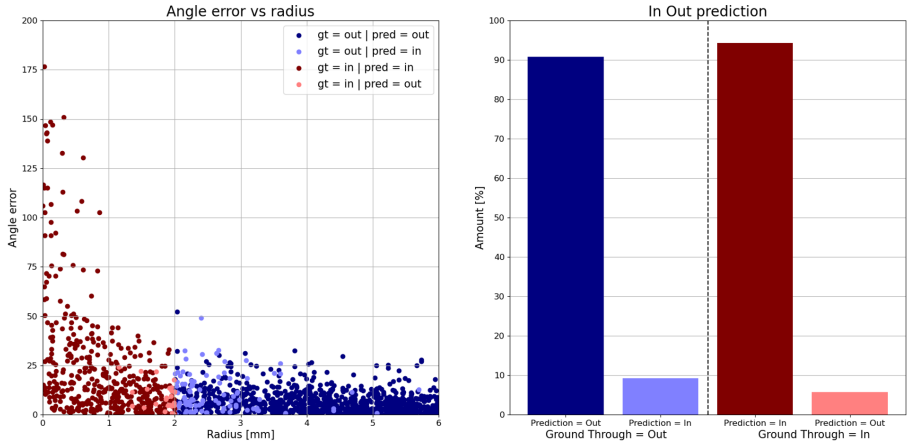


Figure A.44: Angle error vs Radius (left) and In Out prediction (right) for the Medical Pipe task and the Parallel fusion architecture.

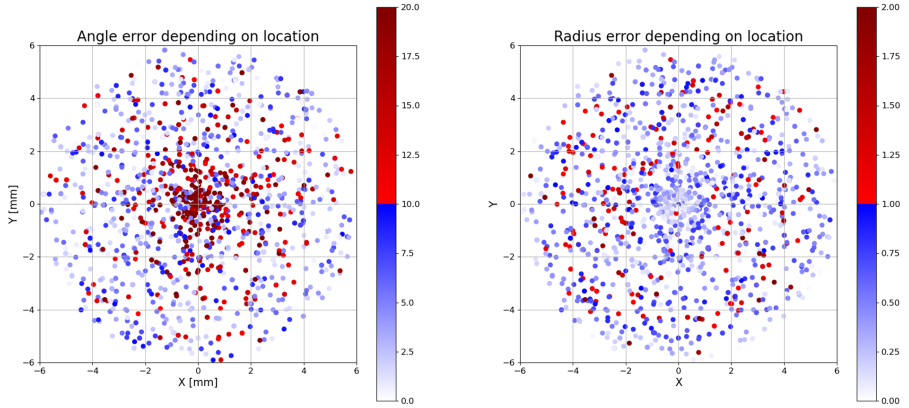


Figure A.45: Angle error (left) and Radius error (right) depending on location for the Medical Pipe task and the Parallel fusion architecture.

Appendix B

Background filtering

In this appendix, the pre-process of filtering the background will be further explained.

During the training process, performance analysis with an activation map revealed that the network exhibited better focus on images with a flat background compared to those with a noisy background, as depicted in Figure B.1.

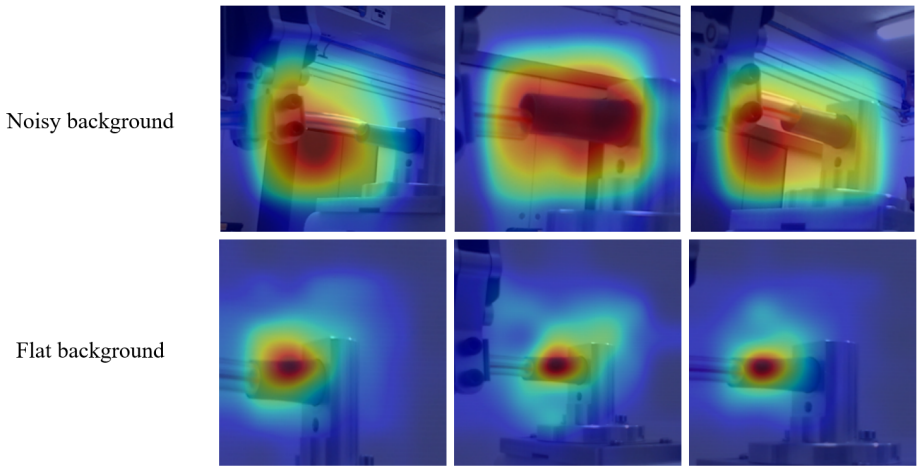


Figure B.1: Activation map of the RGB models train with noisy and flat backgrounds.

Therefore, a method for filtering the background of the RGB images using the depth data was proposed as part of the data pre-processing. The depth data was utilized to generate a binary mask, which was then used to filter the background from the RGB image, as illustrated in Figure 5.6.

To evaluate the pre-process performance the various architectures were trained on two different datasets, one with a noisy background and the other with a filtered background. Samples of these datasets are shown in Figure B.2

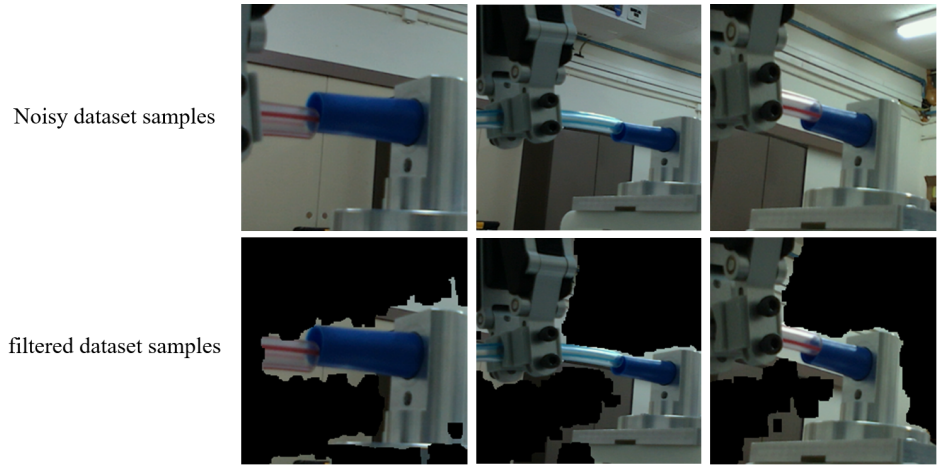


Figure B.2: Samples from the train set with and without the background filtering.

The performance evaluation is based on a test dataset that contains various backgrounds. Samples of these datasets are shown in Figure B.3.

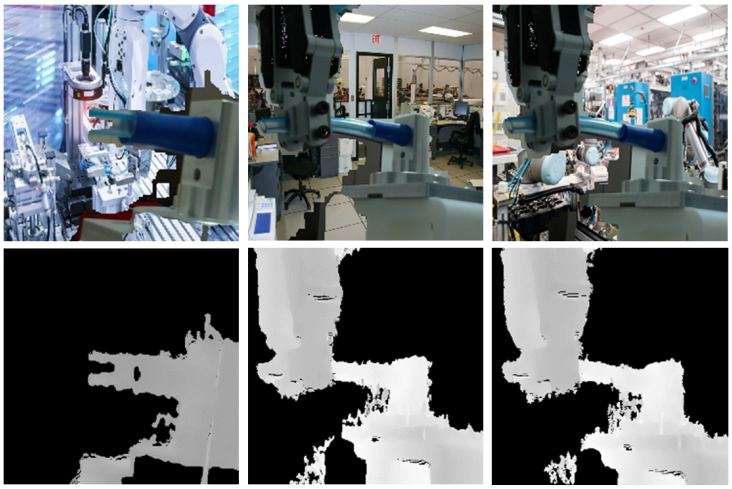


Figure B.3: Samples from the test set of the background filtering evaluation.

The results of the evaluation are presented in Table 6.2. and shows that the best results were obtained for the models trained on the dataset with background filtering, as it is invariant to background changes. Consequently, using background filtering contributes to generalization by enabling the net- work to disregard background changes in the scene.

Appendix C

IMVC poster

The poster in this appendix was prepared based on the research conducted as part of the thesis. It was presented at the IMVC - Israel Machine Vision Conference, which focuses on computer vision technology and includes lectures, presentations, and poster presentations.

This poster encapsulates the essence of our research, containing all its components from the problem statement to the performance analysis.

Estimation of relative position between objects for robotic insertion of LDOs using classification learning methods

TECHNION Israel Institute of Technology

Sher Hazan
Anath Fischer, Ronit Schneor

CAD Laboratory
Mechanical Engineering Faculty

Abstract

This work proposes a robust method for evaluating the relative position between LDOs in real-time assembly tasks. The proposed approach utilizes a classification CNN architecture with RGB-D data fusion. Several fusing approaches were tested and compared. In order to use classification CNN on a continuous problem, the scene space was divided into sub-areas as classes. The classification resulted with the prediction of probabilities of each class. Subsequently, these probabilities were combined using linear interpolation, resulting in an accurate evaluation of the relative position between the objects. For the validation process, three Peg-In-Hole tasks were defined. An RGB-D realistic database was created for each task. Then, pre-processing and augmentation techniques were applied on the data. Finally, the CNNs were trained separately using the same setup to ensure comparable results. The method was evaluated by embedding and testing it in the insertion process of each task at the Technion and the in industry. The feasibility of the proposed method was demonstrated in those tasks, showing an automatic process with high performance.

Problem Statement

Estimating the relative position between objects in PIH tasks.
With a focus on linear deformable objects (LDO).
Using classification learning methods.

Challenges:

Real time operation: 30 [FPS].

Accuracy:

- 1 [mm] in the radius
- 10 [degrees] in the angle.

Robustness: To environment changes and camera viewpoints.

Approach

A pipeline for estimating the relative position between objects using learning method embedded into real robotic cell:

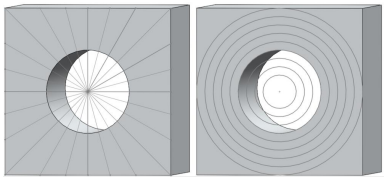
Capturing RGB-D data:

The RGB-D data captured on the robotic cell using RGB-D camera.

Estimation of relative position:

- **Discretization**: In order to use classification methods on a continuous tasks, the scene space was divided into sub-areas as classes.
- **Classification**: Classifying the interaction state between the hole and the object, according to the classes defined. resulting in the prediction of probabilities to be in each class.
- **Estimation**: The probability vector is combined using linear interpolation, resulting in an accurate evaluation of the relative position between the objects, in terms of radius and angle.

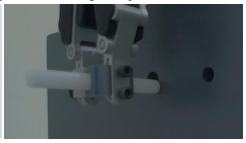
Dividing by angle | Dividing by radius



Test cases

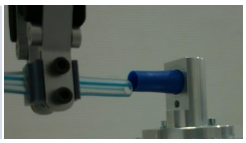
For the validation process of the proposed method, three PIH tasks were defined, varying in their complexity.

Classical Peg in Hole
Rigid – Rigid

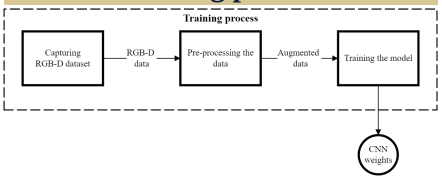


Wiring 1[mm] wire to a connector
Non-Rigid – Rigid

Insert medical pipe to non-rigid connector
Non-Rigid – Non-Rigid



Training process

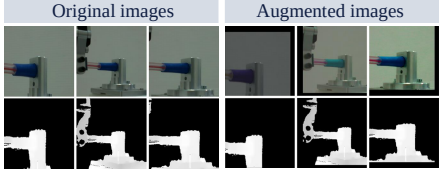


Capturing RGB-D dataset: 2500 RGB-D scenes was captured for each task.

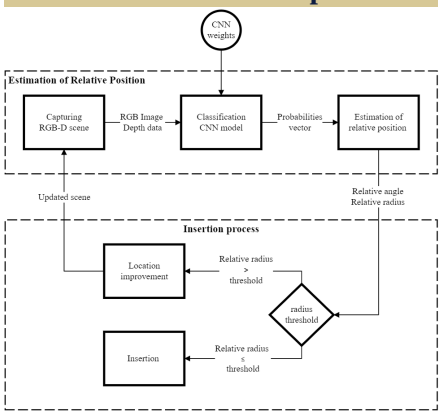
Pre-processing the data: Extract patches of various sizes and shapes from the dataset, to expand it to 7,500 unique scenes. Apply augmentations include lighting and color adjustments as well as translations and rotations.

Training the model: To ensure comparable results, all networks were trained using identical setup and hyperparameters.

Original images | Augmented images



Estimation of relative position



Capturing RGB-D scene: Capturing the RGB-D scene.

Rotated the scene 180 degrees. Resizing and cropping a 240x240 patch to match the required input.

Classification CNN models: For each task, five CNN architectures were designed and trained:

model	Input	Fusion location
RGB	RGB image	-
Depth	Depth data	-
RGB-D early fusion	Concatenated RGB-D	Before input
RGB-D late fusion	Separated RGB-D	Inside the model
RGB-D parallel fusion	Separated RGB-D	After output

Estimation of relative position: The continuous relative position α, r is obtained from the probability vector in three steps:

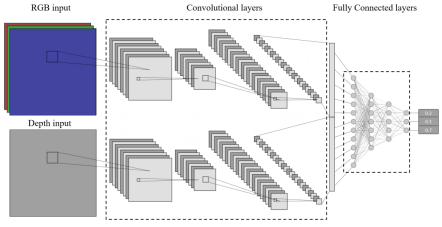
- Thresholding negligible values to suppress noise from the probability vector.
- Normalizing the filtered probability vector.
- Calculation of the mean value across all classes based on the probability vector.

Location improvement: If the relative radius is greater than the threshold The robotic arm improve its location.

Insertion: If the relative radius is smaller then the threshold The robotic arm start the insertion.

The threshold is determined based on the minimum error in the radius, which ensures successful insertion of the object.

Late fusion architecture

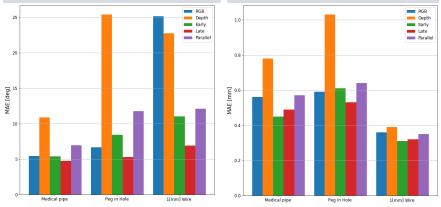


Performance Analysis

The method's effectiveness was evaluated based on the Mean Absolute Error (MAE) across the different models and tasks.

The late fusion method stands out as the most effective overall, consistently producing the lowest errors across most cases.

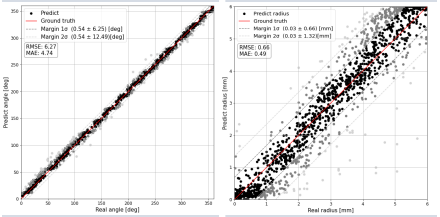
Angular MAE | Radial MAR



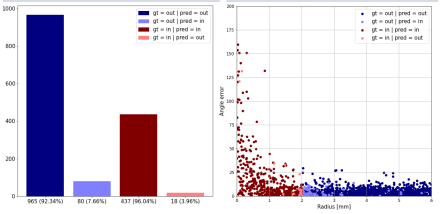
The following graphs represent the results of the late fusion model applied to the medical pipe task.

Medical pipe insertion with late fusion model

Real vs Predicted Radius | Real vs Predicted Angle



In Out prediction | Angle error vs Radius

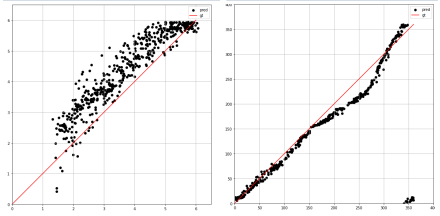


Industrial Application

To validate the real-world applicability and the generalization of the method, a performance test was conducted in the industry. This test involved the task of medical pipe insertion and was carried out without additional training.

Medical pipe insertion at the industry

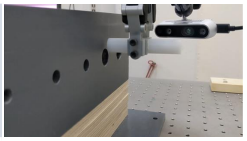
Real vs Predicted Radius | Real vs Predicted Angle



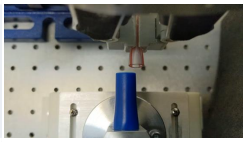
Video Demonstrations

Observe the insertion process in action by scanning the QR code:

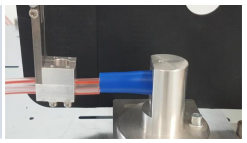
Peg in Hole insertion



Medical Pipe insertion



Medical Pipe Insertion at industry



Contact:

Sher Hazan
sherhazan1115@gmail

References:

[1] Dhruv, P. and Naskar, S., 2020. Image classification using convolutional neural network (CNN) and recurrent neural network (RNN): a review. Machine Learning and Information Processing: Proceedings of ICMLIP 2019, pp.367-381.

[2] Jiang, J., Huang, Z., Bi, Z., Ma, X. and Yu, G., 2020. State-of-the-Art control strategies for robotic PiH assembly. Robotics and Computer-Integrated Manufacturing, 65, p.101894.

[3] Gao, M.; Jiang, J.; Zou, G.; John, V.; Liu, Z. RGB-D-based object recognition using multimodal convolutional neural networks: A survey. IEEE Access 2019, 7, 43110–43136.

[4] Chen, J., Zhang, L., Liu, Y. and Xu, C., 2020, July. Survey on 6D pose estimation of rigid object. In 2020 39th Chinese Control Conference (CCC) (pp. 7440-7445). IEEE.

Appendix D

Learned impedance policies facilitate robotic assembly in general direction and integration of visual sensor

The article in this appendix, titled "Learned Impedance Policies Facilitate Robotic Assembly in General Direction and Integration of Visual Sensor," authored by Eylon Cohen[1] and Sher Hazan[2], is currently under review

In this article, our approach was utilized as a Visual Sensor integrated into the PIH insertion pipeline to enhance the insertion performance.

Learned impedance policies facilitate robotic assembly in general direction and integration of visual sensor

Eylon Cohen[1], Sher Hazan[1], Ronit Schneor[1], Anath Fischer[2] and Miriam Zacksenhouse[2] *Member, IEEE*

***Abstract*—Impedance control has been demonstrated to facilitate robotic assembly under uncertainties. Position based impedance control, also known as admittance control, involves modifying the reference trajectory in response to force and torque (F/T) measurements, to satisfy the desired dynamical behavior. However, previous work focused on cases when the F/T sensor is co-aligned with the direction of insertion. In those cases, when the mating parts overlap, the direction of the torque vector provides unambiguous information about the relative location of the parts. However, this relationship does not hold when the F/T sensor is not co-aligned with the direction of insertion. Here we propose a method for impedance control independent of the alignment between the F/T sensor and the direction of insertion. The method is based on transforming the F/T measurements to a virtual location that further simplifies the impedance control, and on replacing the noisy friction forces with virtual friction forces. The proposed method facilitates the integration of learning-based visual sensor when a wrist-mounted camera is available. The impedance parameters were learned in simulation using reinforcement learning (RL). The proposed method, with the learned parameters, was evaluated both in stimulation and on a physical robot (UR5e). The experiments demonstrate that the proposed method performs successfully despite uncertainties, and generalizes well to different sizes and even to assembly of flexible parts. Finally, the visual sensor improved performance up to 100% success rate on the physical robot. Thus, the proposed method can facilitate the integration of autonomous robots in industrial assembly.**

***Index Terms*—Compliance control, impedance control, robotic control, robotic assembly, machine learning, reinforcement learning, force and torque sensor.**

I. Introduction

ROBOTS are robust, precise, programmable, and repeatable machines that can be designated for a large variety of tasks [1]. Hence, there has been a growing interest in transferring tasks currently performed by humans to autonomous robots [2]. Indeed, over the last few decades robots have been integrated into a range of industries, especially large industries (25% of Europe's large enterprises use robots [3]).

One of the main challenges for robots is the execution of contact-rich assembly tasks, since they sensitive to uncertainties in the pose of the parts. Nevertheless, assembly tasks are common and necessary in a wide range of industries, including medical devices, heavy industry, and electronics.

A common approach to overcome uncertainties is to control the position of the mating parts precisely, but, in many cases, this is not cost-effective. Hence, assembly tasks are mainly performed by humans (only 2% of Europe enterprises use robots for assembly tasks [3]). Robots that could overcome uncertainties in assembly tasks are expected to increase production rate, lower costs, and significantly reduce the risk of human injury.

Impedance control is a well-known control strategy for tasks involving interaction with the environment [4]. Position based impedance contorl, also known as admittance control, modifies the reference trajectory in response to force and torques (F/T) measurements, according to the desired dynamical behavior. However, the application of impedance control when the F/T sensor is not co-aligned with the direction of insertion is challenging, as explained in Section III-B.

Here we address the challenge of performing assembly tasks despite uncertainties, when the F/T sensor is not co-aligned with the direction of insertion. Furthermore, we address the following challenges: (1) generalization over sizes, (2) generalization to assembly of flexible parts, and (3) integration of visual information to improve performance when a wrist camera is available.

The proposed method relies on transforming the measurements of an actual F/T sensor to a virtual location. That location is selected so the resulting virtual torque can be converted directly to virtual friction forces, which facilitate impedance control. We use position-based impedance control and learn the parameters of the impedance matrices using reinforcement learning (RL).

The work was motivated by the challenge of horizontal assembly of flexible medical tubes of $8.5[mm]$ diameter (MD industries [5]). To avoid the simulation of flexible tubes, training was based on the simulation of assembly of rigid parts of similar sizes. Using the learned parameters, the proposed methods (with and without a visual sensor) were evaluated on a physical robot, UR5e, with both rigid parts and flexible medical tubes (see video).

*The work was supported by the Israel Innovation Authority as part of the ART (Assembly by Robotic Technology) consortium (grant number 74885).

[1] Eylon Cohen, Sher Hazan, and Ronit Schneor are with the Faculty of Mechanical Engineering, Technion, Israel. ela3135@gmail.com, sherhazan1115@gmail.com, schneor@me.technion.ac.il

[2]Anath Fischer and Miriam Zacksenhouse are with the Faculty of Mechanical Engineering, Technion, Israel and the Technion Autonomous Systems Program. meranath@technion.ac.il, mermz@technion.ac.il

II. Related Work

Impedance control endows the end-effector (EEF) with the desired impedance, e.g., stiffness, damping and inertia, to achieve the desired trade-off between position deviations and force deviations during interactions with the environment. However, to date, the applications of impedance control to assembly tasks focused mainly on cases when the F/T sensor is co-aligned with the direction of insertion ([6], [7], [8], [9]). In those cases, when the mating parts overlap, the direction of the resulting torque depends on the direction of the overlap [10], and thus can be used to determine the required correction.

However, when the F/T is not co-aligned with the direction of insertion, the direction of the torque vector does not depend on the relative location of the parts, as further explained in Section III-B. Nevertheless, this case is common in daily life and industrial applications due to spatial arrangements and space limitations.

In robotic applications impedance control is usually implemented in software rather than in hardware. The standard dynamic based impedance control introduced by Hogan [11] relies on an accurate dynamic model of the robot [12], [13] and thus may hamper sim2real. Instead, we implemented position based impedance control, also known as admittance control, which modifies the reference trajectory in response to F/T measurements [13], [14].

Following our previous work on vertical assembly using impedance policies [8], [9], we learn a fixed set of impedance parameters using RL. However, while our previous work demonstrated the importance of using non-diagonal and especially asymmetric impedance matrices, here we use diagonal matrices. The diagonal matrices are sufficient in the proposed method since we introduce virtual friction forces, which account for the effect of torques, as further explained in Section III-C. Thus, the proposed method greatly reduces the number of free parameters that have to be learned by RL.

III. Reinforcement Learning of Impedance Policies

A. Position Based Impedance Control

Position based impedance control modifies the desired trajectory X^w_{dw}, specified by the position vector $p^{w\ 3\times1}_{dw}$ and rotation matrix $R^{w\ 3\times3}_{dw}$ with respect to the world frame, to a compliant trajectory X^w_{cw}, specified by $p^{w\ 3\times1}_{cw}$ and $R^{w\ 3\times3}_{cw}$. Using the angle/axis representation of the relative orientation of the compliant and desired trajectories [15], and assuming small relative angle, the 6 degrees of freedom impedance model can be expressed as:

$$M^{6\times6}\ddot{x}^{d\,6\times1}_{cd} + C^{6\times6}\dot{x}^{d\,6\times1}_{cd} + K^{6\times6}x^{d\,6\times1}_{cd} = F^{d\,6\times1} \quad (1)$$

where: $x^d_{cd} = \left[{p^d_{cd}}^T, {o^d_{cd}}^T\right]^T$ is the relative position p^d_{cd} and the small relative orientation o^d_{cd} of the compliant trajectory with respect to the desired trajectory (expressed in the frame attached to the desired trajectory), $\dot{x}^d_{cd} = \left[{p^d_{cd}}^T, {\omega^d_{cd}}^T\right]^T$ is the translation and angular velocities between the trajectories, expressed in the frame attached to the desired trajectory, $M, C, K \in R^{6\times6}$ are the desired inertia, damping and stiffness of the impedance model, respectively, and $F^{d\ 6\times1}$ is the force and torque vector.

Eq. (1) can be expressed in state space representation:

$$\dot{X}^d_{cd} = AX^d_{cd} + BU^d \quad (2)$$

where:

$$X^{d\ 12\times1}_{cd} = \begin{bmatrix} x^d_{cd} \\ \dot{x}^d_{cd} \end{bmatrix} = \begin{bmatrix} p^d_{cd} \\ o^d_{cd} \\ \dot{p}^d_{cd} \\ \omega^d_{cd} \end{bmatrix} \ \& \ \dot{X}^{d\ 12\times1}_{cd} = \begin{bmatrix} \dot{p}^d_{cd} \\ \omega^d_{cd} \\ \ddot{p}^d_{cd} \\ \dot{\omega}^d_{cd} \end{bmatrix}, \quad (3)$$

$$A^{12\times12} = \begin{bmatrix} 0^{6\times6} & I^{6\times6} \\ -M^{-1\ 6\times6}K^{6\times6} & -M^{-1\ 6\times6}C^{6\times6} \end{bmatrix}, \quad (4)$$

$$B^{12\times6} = \begin{bmatrix} 0^{6\times6} \\ M^{-1\ 6\times6} \end{bmatrix} \ \& \ U^{d\,6\times1} = F^{d\,6\times1} \quad (5)$$

Finally, since realistic control executes in discrete time steps of ΔT, a discrete state space realization is implemented:

$$X^d_{cd}(k+1) = A_d X^d_{cd}(k) + B_d U^d(k) \quad (6)$$

where:

$$A_d = e^{A\Delta T} \ \& \ B_d = A^{-1}(A_d - I)B \quad (7)$$

To assure stability we verified that the eigenvalues of A_d are inside the unit circle.

The desired trajectory X^w_{dw} was designed as a minimum jerk trajectory to the desired final position and orientation. The compliant trajectory in the world frame, X^w_{cw}, was computed from the desired trajectory and the deviation X^d_{cd}.

B. Virtual Sensor

When the peg partially overlaps the hole, the reaction forces are skewed to one side of the peg. If the F/T sensor is co-aligned with the direction of the insertion, as in Figure 1(a), the direction of the resulting torque depends on the relative position of the peg and hole [10]. In that case, the non-diagonal terms of the stiffness matrix can help determining the necessary correction movements [9]. However, when the F/T sensor is not co-aligned with the direction of insertion, as in Figure 1(b), the direction of the torque may not vary with the relative location of the peg and hole.

To overcome this problem, this study suggests to transform the measured F/T to a virtual location, thus generating a virtual sensor (Figure 1(c)). In general, the virtual sensor can be located in any place, but, as will be justified latter, we suggest to position it at the center of the edge of the peg, facing the hole, as indicated in Figure 1(c).

In order to derive the transformation between the measured and virtual F/T sensors, we first derive the relationship between the measured and external F/T. For the static and gravity-free case, the force, F_S and torque T_S, measured by

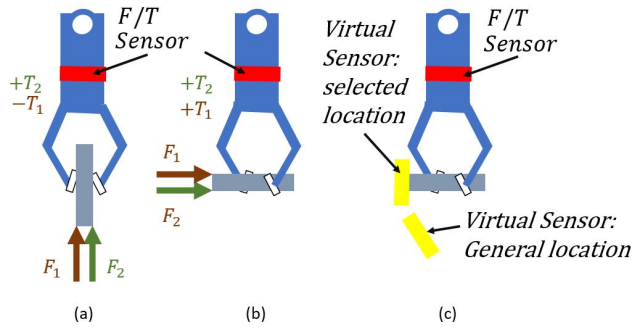


Fig. 1: Effect of the alignment of the F/T sensor with the direction of insertion in a 2-dimensional case ((a) and (b)) and possible locations of the virtual F/T sensor (c). Co-aligned sensor (a): the sign of the torque depends on the side of the peg on which the reaction force is applied. Non co-aligned sensor (b): the sign of the torque is the same independent of where the reaction force is applied. Virtual sensor can be located in different places, including the selected place at the center of the the edge of the peg facing the hole (c).

the F/T sensor located at P_S are related to the external force F_{ext} that acts at P_{ext} and to the external torque T_{ext} by:

$$\begin{bmatrix} F_S \\ T_S \end{bmatrix} = \begin{bmatrix} F_{ext} \\ (P_{ext} - P_S) \times F_{ext} + T_{ext} \end{bmatrix} \tag{8}$$

Similarly, a virtual sensor in some virtual position P_V would have measured the force F_V and torque T_V given by:

$$\begin{bmatrix} F_V \\ T_V \end{bmatrix} = \begin{bmatrix} F_{ext} \\ (P_{ext} - P_V) \times F_{ext} + T_{ext} \end{bmatrix} \tag{9}$$

Given that:

$$P_{ext} - P_V = (P_S - P_V) + (P_{ext} - P_S), \tag{10}$$

Eq. (9) can be expressed as:

$$\begin{bmatrix} F_V \\ T_V \end{bmatrix} = \begin{bmatrix} F_{ext} \\ ((P_S - P_V) + (P_{ext} - P_S)) \times F_{ext} + T_{ext} \end{bmatrix} \tag{11}$$

Finally, inserting Eq. (8) in Eq. (11) results in:

$$\begin{bmatrix} F_V \\ T_V \end{bmatrix} = \begin{bmatrix} F_S \\ (P_S - P_V) \times F_S + T_S \end{bmatrix} \tag{12}$$

Eq. (12) implies that, for the static and gravity-free case, the F/T measured by the actual sensor can be transformed to the F/T measured at any virtual sensor given the relative position of the actual and virtual sensors. For the general case where the body's dynamic and gravity are not negligible, it can be shown [16] that those elements can be compensated in real-time.

For reasons detailed in the next section, we placed the virtual sensor at the center of the bottom surface of the peg. The position of the virtual sensor was determined in a series of calibration experiments, and possible calibration errors were considered in the training process as detailed in Section III-E.

C. Virtual Friction Forces

We suggest to locate the virtual sensor such that the friction forces, which tend to by noisy, do not affect the virtual torque. Hence, we locate the virtual sensor at the center of the edge of the peg facing the hole and align its z-axis with the direction of insertion. In that case, the vector $(P_S - P_V)$ lies on the same plane as the friction Forces, and thus the x and y components of the virtual torque, T_{V_x} and T_{V_y}, are not affected by them.

When the peg contacts the surface of the hole (see Figure 2), the 2-dimensional vector R from the virtual sensor to the location of the equivalent reaction force F_{V_z}, is given by:

$$R = \begin{bmatrix} -\frac{T_{V_y}}{F_{V_z}} & \frac{T_{V_x}}{F_{V_z}} \end{bmatrix}^T \tag{13}$$

As is evident from Figure 2, under contact, the vector R should be along the line connecting the centers of the hole and peg. Hence, we suggest to replace the measured friction forces, which tend to be noisy, with an equivalent unit virtual friction force $\hat{F} = -\frac{R}{|R|}$. This replacement is expected to facilitate the ability of the impedance controller to modify the reference trajectory in the direction of the hole center.

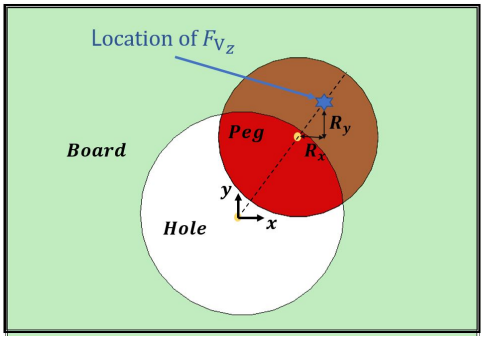


Fig. 2: Illustration of the contact surface between the peg and the hole. The blue star represents the location of the equivalent reaction force F_{V_z} acting on the peg and R represents its position relative to the center of the peg. Ideally, it is assumed that R is on the line connecting the centers of the peg and hole.

Thus, the right hand side of the impedance model (Eq. (1)) is replaced by $Q^{6\times1} = [\hat{F}, F_{V_z}, T_{V_x}, T_{V_y}, T_{V_z}]^T$, resulting in the control method illustrated in Figure 3.

D. Action space

In the RL framework, the agent interacts with its environment by performing actions. As in our previous work [9], the action in our method determines the parameters of the impedance matrices K, C, M, which in turn affects the trajectory correction. In [9] it was demonstrated that it is sufficient to learn space-invariant impedance parameters. Thus, here we focus on learning a fixed set of parameters.

We hypothesized that given the virtual friction forces, it would be sufficient to use diagonal impedance matrices rather than the asymmetric matrices used in [9]. As explained

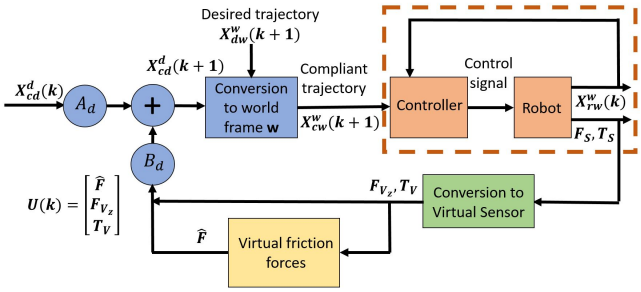


Fig. 3: High-level control scheme: position based impedance control modifies the desired trajectory X^w_{dw}, specified in the world frame, w, to a compliant trajectory X^w_{cw}, such that its deviation from the desired trajectory, X^d_{cd}, satisfies the impedance model (captured by A_d and B_d) in response to the virtual F/T measurements (F_V and T_V) and virtual friction forces ($\hat{F}$). The compliant trajectory is the input to the low-level controller marked in a dashed orange frame, that also receives the robot state X^w_{rw}. The virtual F/T measurements are computed from the actual F/T measurements (F_S and T_S). The subscripts d, c, w and r refer to the desired, compliant, world and robot, respectively

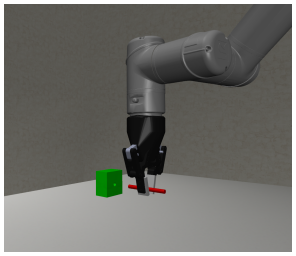


Fig. 4: Simulation of the horizontal insertion task showing a close up of the gripper, the peg and the board with a hole.

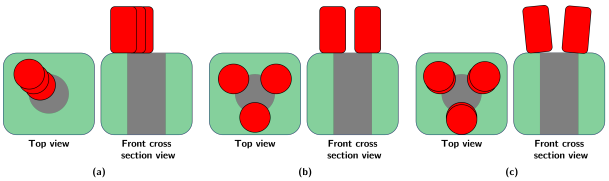


Fig. 5: Types of possible errors that were deliberately introduced in the simulation to train a policy that is robust to uncertainties. Top view (right) and front cross-section view (left) of radial errors (a), angular error (b), and peg orientation errors (c).

and demonstrated in [9], asymmetric impedance matrices are important in order to modify the x and y position in response to T_y and T_x, respectively. However, the virtual friction force $\hat{F}$, which is a unit vector in the direction of R given by Eq. (13), reflects already the effect of the torque. Thus, we restricted the impedance matrices to be diagonal.

Finally, when the parts are axisymmetric (i.e., have round cross-section), the structure of each impedance matrix can be further restricted to:

$$*^{6\times6} = diag(\begin{bmatrix} *_{xy} & *_{xy} & *_z & *_{\theta_x\theta_y} & *_{\theta_x\theta_y} & *_{\theta_z} \end{bmatrix}) \quad (14)$$

where $*$ stands for K, C or M. Thus, each matrix is defined by 4 parameters, for a total of only 12 free parameters.

E. Learning the impedance policy in Simulation

The simulation environment was implemented using robosuite [17], a well-known, open-source, physical simulation platform powered by the MuJoCo [18] physical engine. The impedance policy, which determines the parameters of the impedance matrices, was trained to perform the nominal task specified below, using Proximal Policy Optimization (PPO) algorithm [19], implemented in Stable baseline 3 [20].

Nominal task: The policy was trained to perform horizontal insertion of a circular peg with diameter $D_{peg} = 8[mm]$ into a round hole with diameter $D_{hole} = 10[mm]$ located on a board mounted vertically on a table, as depicted in Figure 4. The peg was held by the gripper so it was horizontal to the table. For simplicity, the location of the board was fixed for all episodes (as explained above, and tested on the physical robot, this does not hamper the generalization to other locations).

In facilitate robustness to uncertainties in the location of the hole and orientation of the peg, and to calibration errors (see end of Section III-B), the following errors were deliberately introduced during training:

- Radial error R_{err} and angular error A_{err} between the actual and presumed locations of the hole (Figure 5 (a) and (b), respectively) were uniformly distributed in the ranges $[1.5, 3.5]$[mm] and $[0, 2\pi]$[rad], respectively
- Peg orientation error O_{err} (Figure 5 (c)) was uniformly distributed in the range $[-5°, 5°]$.
- Axial and radial calibration errors in the position of the virtual sensor, A_{cal} and R_{cal}, respectively, were uniformly distributed in the ranges $[-5, 5]$[mm] and $[0, 2]$[mm,] respectively.

Reward function: The reward was composed of two parts: (1) A step-by-step penalty that depended on the relative location and orientation of the peg and hole, and (2) A final bonus, $R_B = 10^5$, that was given for successful insertions

F. Visual Friction Forces

A wrist camera can be used to improve performance by estimating the direction of the virtual friction forces visually. For that purpose, we defined the angle α, between the vector from the center of the hole to the center of the peg and the x-axis. The angle α was estimated from the visual sensor and used to define the visual friction force $\hat{F} = -\begin{bmatrix} cos(\alpha) & sin(\alpha) \end{bmatrix}^T$, which was used instead of the virtual friction force.

The relative angle α was estimated using a classification-based approach. First, the spatial location was discretized into several angular ranges according to the desired resolution (here we used 16 ranges). Second, a ResNet-18 [21] network was trained to classify the angular range. Finally, the output

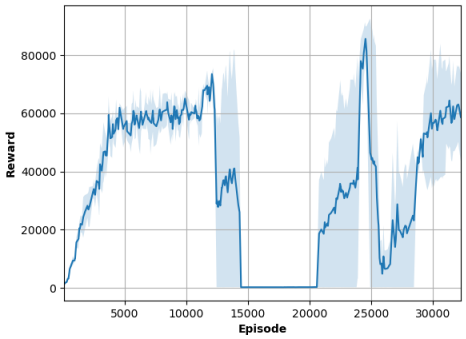


Fig. 6: Average reward during a training session of horizontal insertion with random errors. Blue background denotes the standard deviation envelope.

of the network was interpreted as the likelihood of each range, and α was estimated using linear interpolation.

IV. Results

A. Simulation Results

Figure 6 depicts the average episode reward during a typical training session described in Section III-E. Recall that the maximum possible reward is $R_B = 10^5$. While the reward increased initially, it dropped sharply before increasing again to its maximum level. This might be attributed to the drastic effect of a slight change in the policy on the impedance.

The performance of the proposed method, with the parameters that resulted in the highest reward during training, was evaluated in simulation on both the nominal task and on two new tasks. The new tasks evaluated generalization over size. Each task was evaluated with and without calibration errors (R_{cal} and A_{cal}) for 200 trials, and the resulting success rates and their confidence intervals (CI) are summarized in Table I.

Table I indicates that the estimated success rates remained above 91% for all tasks, thus demonstrating robustness to uncertainties and generalization over size. Interestingly, if the location of the virtual sensor is accurate ($R_{cal} = A_{cal} = 0$), the estimated success rate of inserting a 4.5[mm] peg into a 6[mm] hole was 100%, despite uncertainties in the radial position of the hole (uniformly distributed in the range of $1-2.5$[mm]). The considered calibration errors and uncertainties in pole orientation degraded performance by $3.5-7.5\%$.

B. Simulation to Real Transfer (Sim2Real)

The proposed method for horizontal insertion, with the parameters learned in the simulation, was implemented on a real robot, UR5e, equipped with an OnRobot HEX-E F/T sensor. Performance on the real robot was evaluated for the same sizes and uncertainties in hole position as in the simulations. Uncertainties in the position of the virtual sensor and the orientation of the peg resulted naturally from the estimation of the location of the virtual sensor and from

D_{peg} [mm]	D_{hole} [mm]	R_{err} [mm]	A_{cal}/R_{cal} [mm]	O_{err} [°]	Success rate [%]	95% CI [%]
8	**10**	**[1.5,3.5]**	0/0	0	97	[93.6,98.6]
			[-5,5]/[0,2]	**[-5,5]**	**93.5**	**[89.2,96.2]**
16	20	[2.5,8]	0/0	0	94	[89.8,96.5]
		[2.5,5]	[-5,5]/[0,2]	[-5,5]	91	[86.2,94.2]
4.5	6	[1,2.5]	0/0	0	100	[98.1,100]
			[-2,2]/[0,1]	[-5,5]	92.5	[88,95.4]

TABLE I: Simulation performance of the trained policy. Success rates and confidence intervals (CI) were estimated from 200 trials of the indicated horizontal insertion task. D_{peg} and D_{hole} are the diameters of the peg and hole, respectively. Errors were uniformly distributed in the indicated ranges in the radial position of the hole R_{err}, peg orientation O_{err} and axial and radial position of the virtual sensor A_{cal} and R_{cal}, respectively. Performance of the nominal task is marked in bold (second row).

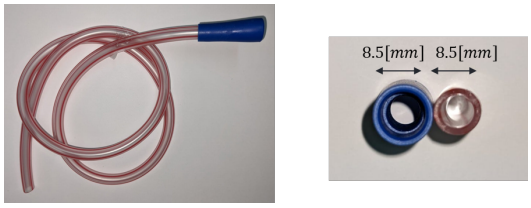


Fig. 7: Flexible medical tubes that were inserted horizontally during experiments on the real robot (MDC industries).

grasping the peg. The experimental results are summarized in the first 3 rows of Table II. The transfer to the real robot degraded performance moderately (by $10.5\% - 11\%$) for the nominal and large pegs (first two rows), and drastically (by 41.5%) for the small peg (third row). This degradation can be attributed to measurement noise in the real sensor and possibly larger calibration errors (in the position of the virtual sensor).

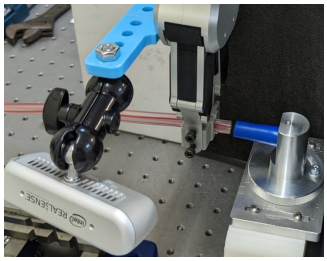
We also tested the generalization of the method to assembly of flexible parts, and in particular, to the assembly of flexible medical tubes (MDC industries [5]). The diameter of both mating tubes was $8.5[mm]$, as depicted in Figure 7. As reported in the last row of Table II (marked in blue), the success rate in assembling the flexible tubes with the real robot was 80%, despite uncertainties in the relative position and the flexibility of the parts.

C. Optional Visual Sensor

A REALSENSE D435 camera was mounted on the gripper of the UR5e, as shown in Figure 8. Separate sets of training images were collected with rigid parts and flexible tubes. The gripped part was placed at uniformly distributed angles and radii around the fixed part: radius of up to $(D_{hole} + D_{peg})/2$ for the rigid parts, and $1.5-6$[mm] for the tubes. Separate networks were trained for the rigid parts and flexible tubes. The trained networks predicted α with mean absolute error (MAE) of $5.98°$ for the rigid parts and $7.33°$ for the flexible tubes (root mean square, RMS, of $8.5°$ and $11.5°$, respectively).

D_{peg} [mm]	D_{hole} [mm]	R_{err} [mm]	Success rate [%]	95% CI [%]
8	**10**	**[1.5, 3.5]**	**83**	**[74.5, 89.1]**
16	20	[2.5, 8]	80	[71.1, 86.7]
4.5	6	[1, 2.5]	51	[41.3, 60.6]
8.5	8.5	[1.5, 2.5]	80	[71.1, 86.7]

TABLE II: Experimental performance of the trained policy, with virtual friction forces, on a physical robot UR5e equipped with OnRobot F/T sensor. Success rates and confidence intervals (CI) were estimated from 100 trials of the specified horizontal insertion task. See Table I for other symbols. Performance of the nominal task is marked in bold. Performance with flexible tubes is marked in blue (last row).



(a)

(b)

Fig. 8: Camera mounted on the end effector during insertion of medical tubes (a) and rigid cylindrical peg (b).

The performance of the impedance controller with both the virtual and visual sensors, was evaluated on the physical robot, as summarized in Table III. Perfect success rates of 100% were achieved for the nominal task on which training was conducted, and even for the larger peg and hole (16[mm] peg into 20[mm] hole), over a large range of position uncertainties (larger than the range used in training the impedance policy). The success rate for inserting the 8.5[mm] medical tube was 98%, with radial uncertainties of 2[mm] to 6[mm]. Comparing these results to the ones to without the visual sensor (Table II), it is evident that integrating the visual sensor increased the success rate over a broader range of uncertainties.

V. Conclusion

The goal of this research was to develop a robust method for peg-in-hole insertion tasks, regardless of whether the direction of insertion is co-aligned with the F/T sensor. In particular, the work focused on horizontal insertion with a vertically aligned gripper. Given the importance of impedance control for contact-rich tasks, and previous success in our laboratory, we developed a method based on impedance control. To handle cases where the insertion axis is not co-aligned with the axis of the F/T sensor, we introduced a virtual sensor located at the center of the edge of the peg, facing the hole. This facilitated the computation of virtual friction forces from the measured torques, instead of the inherently noisy friction forces. The virtual friction force directly relates the torques to linear path corrections and so even diagonal impedance matrices with only 12 free parameters worked well.

D_{peg} [mm]	D_{hole} [mm]	R_{err} [mm]	Success rate [%]	95% CI [%]
8	**10**	**[1.5, 10]**	**100**	**[96.3, 100]**
16	20	[4, 14]	100	[96.3, 100]
4.5	6	[1, 3]	90	[82.6, 94.5]
8.5	8.5	[2, 6]	98	[93, 99.4]

TABLE III: Experimental performance of the trained policy, with visual friction forces, on a physical robot UR5e equipped with OnRobot F/T sensor and REALSENSE D435 camera. Success rates and confidence intervals (CI) were estimated from 100 trials of the specified horizontal task. See Table I for other symbols. Performance of the nominal task, with larger uncertainties, is marked in bold. Performance with the flexible tubes is marked in blue (last row).

RL was used to learn the impedance parameters for a nominal horizontal insertion of rigid parts, despite uncertainties in the position. Simulation results indicate good success rate (93%) for the nominal task, and good generalization across size, especially to larger sizes. The transition from simulation to the physical robot degraded performance by about 10% for the nominal task.

The method generalized very well to flexible pegs, and achieved a good success rate (80%) for inserting $8.5[mm]$ medical tubes. The generalization to flexible tubes is especially important since flexible parts are difficult to simulate well. Our method demonstrate that it is enough to learn the impedance policy in simulations of rigid parts of similar sizes - and the learned policy perform well on flexible parts. Nevertheless, future work may evaluate the effect of retraining on the real robot.

We have shown that the method can be easily enhanced using a visual sensor when available by computing visual, rather than virtual, friction forces. Using the visual sensor, success rates on the real robot increased to 100% for the nominal task, and for larger sizes, and to 98% for assembly of flexible medical tubes, despite larger ranges of uncertainties.

In conclusion, the proposed method facilitates robotic assembly in general directions, independent of the axis of the F/T sensor, and achieves good success rates despite position uncertainties. It generalizes well to larger sizes, and most importantly, to assembly of flexible parts like medical tubes. Finally, the proposed method can take advantage of a wrist-mounted visual sensor and achieve close to 100% success rate. Thus, the proposed method can facilitate the integration of autonomous robots in industrial assembly.

REFERENCES

[1] T. Turja and A. Oksanen, "Robot acceptance at work: a multilevel analysis based on 27 eu countries," *International Journal of Social Robotics*, vol. 11, no. 4, pp. 679–689, 2019.

[2] G. Michalos, S. Makris, J. Spiliotopoulos, I. Misios, P. Tsarouchi, and G. Chryssolouris, "Robo-partner: Seamless human-robot cooperation for intelligent, flexible and safe operations in the assembly factories of the future," *Procedia CIRP*, vol. 23, pp. 71–76, 2014.

[3] P. E. News, "25% of large enterprises in the eu use robots," https://ec.europa.eu/eurostat/web/products-eurostat-news/-/ddn-20190121-1, 2019.

[4] N. Hogan, "Impedance control: An approach to manipulation: Part ii—implementation," 1985.

[5] Mdc, "Mdc industries," http://www.mdcindustries.com, 2022.

[6] S. Chan and H. Liaw, "Generalized impedance control of robot for assembly tasks requiring compliant manipulation," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 4, pp. 453–461, 1996.

[7] J. F. Broenink and M. L. Tiernego, "Peg-in-hole assembly using impedance control with a 6 dof robot," in *Proceedings of the 8th European Simulation Symposium*. Citeseer, 1996, pp. 504–508.

[8] O. Spector and M. Zacksenhouse, "Deep reinforcement learning for contact-rich skills using compliant movement primitives," *arXiv preprint arXiv:2008.13223*, 2020.

[9] S. Kozlovsky, E. Newman, and M. Zacksenhouse, "Reinforcement learning of impedance policies for peg-in-hole tasks: Role of asymmetric matrices," *IEEE Robotics and Automation Letters*, 2022.

[10] T. Kusakabe, S. Sakaino, and T. Tsuji, "Design of non-diagonal stiffness matrix for assembly task," *arXiv preprint arXiv:2210.16594*, 2022.

[11] N. Hogan, "Impedance Control Part1-3," *Transaction of the ASME, Journal of Dynamic Systems, Measurement, and Control*, vol. 107, no. March 1985, pp. 1–24, 1985.

[12] A. Lu, Z.; Goldenberg, "Implementation of Robust Impedance and Force Control," pp. 145–163, 1992.

[13] T. Valency and M. Zacksenhouse, "Accuracy/robustness dilemma in impedance control," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 125, no. 3, pp. 310–319, 2003.

[14] M. Schumacher, J. Wojtusch, P. Beckerle, and O. von Stryk, "An introductory review of active compliant control," *Robotics and Autonomous Systems*, vol. 119, no. April 2020, pp. 185–200, 2019.

[15] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Six-dof impedance control based on angle/axis representations," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 289–300, 1999.

[16] J. Duan, Z. Liu, Y. Bin, K. Cui, and Z. Dai, "Payload identification and gravity/inertial compensation for six-dimensional force/torque sensor with a fast and robust trajectory design approach," *Sensors*, vol. 22, no. 2, p. 439, 2022.

[17] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," in *arXiv preprint arXiv:2009.12293*, 2020.

[18] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," 2019.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

בכדי לאמן את הרשתות, נבנה dataset ראליסטים הכוללים 2500 תמונות צבע ומידע עומק בעבור כל משימה. הסצנות צולמו באמצעות מצלמת תלת מימד בתא רובוטי בטכניון. לאחר מכן ביצענו עיבוד מקדים על התמונות, הן בשביל להגדיל את ה - dataset והן בכדי לקבל הכללה טובה יותר. לבסוף כל הארכיטקטורות אומנו עם אותם ההגדרות והפרמטרים בכדי שניתן יהיה להשוות בין התוצאות.

השיטה הוערכה על ידי הטמעתה ובדיקתה בתהליך ההחדרה הרובוטית הכולל את השלביים הבאים:

- חיזוי וקטור ההסתברויות באמצעות הרשת.
- שערוך המיקום היחסי בין האובייקט המוחדר לחור.
- שימוש במיקום היחסי בכדי לשפר את מיקום האובייקט המוחדר.

התהליך חוזר על עצמו עד שהאובייקט מקביל לחור, ברגע שמצב זה קורה תהליך ההחדרה מתחיל.

לסיכום, בעבודת מחקר זו הצגנו פתרון מקצה לקצה לבעיית חיווט אובייקטים גמישים באמצעות שיטות למידה עמוקה וניתוח תמונה. השיטה המוצעת מאפשרת אוטומציה וחישוב בזמן אמת של מיקום יחסי בין אובייקטים גמישים עבור משימות שונות. מהתוצאות של השטיה שפיתחנו ניתן הן להסיק האם האובייקטים במצב יחסי הרצוי והן לחשב את המרחק ממצב זה (רדיוס וזווית).

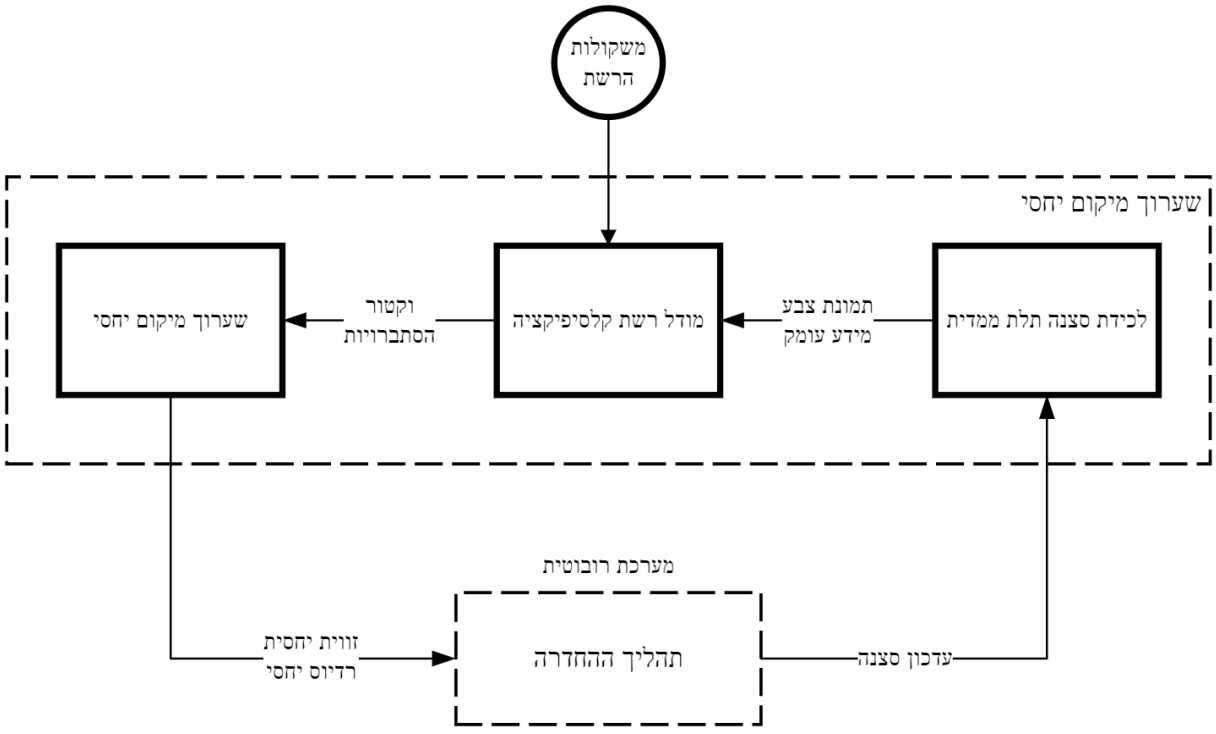
היתכנות השיטה שפותחה נבדקה גם בתעשייה על אובקטים בסביבה אמתית ומורכבת, במסגרת שיתוף פעולה עם מאגד של הראשות לחדשנות.

משום שעל השיטה לעבוד בעולם האמיתי בו יש לבצע פעולות בזמן אמת וקשה לאסוף מידע לאימון, בחרנו לבסס את השיטה שפתחנו בהתבסס על רשת Resnet-18 [8]. זאת מכיוון שרשת זו רצה בזמן אמת, דורשת מעט נתונים לאימון, וניתן להשתמש בה הן בסיווג והן ברגרסיה. את הרשת הרחבנו לקבלת נתונים צבע ועומק בשיטות היתוך שונות.

השיטה כללה מספר שלבים:

- דיסקרטיזציה - על מנת להשתמש בשיטות סיווג במשימות רציפה בוצעה דיסקרטיזציה לבעיה על ידי חלוקת המרחב למחלקות (זווית ורדיוס).
- סיווג - נעשה שימוש ברשת עמוקה לשם סיווג מצב האינטראקציה האובייקטים על פי המחלקות המוגדרות בחלוקות המרחב. כאשר הקלט של הרשת הינה תמונות צבע ומידע עומק והפלט הוא ההסתברות להימצא בכל אחד מהמחלקות במרחב.
- שערוך - ביצוע אינטרפולציה ליניארית על הפלט של הרשת כדי לשערך את הערך הרציף של המיקום היחסי.

לאחר שערוך המיקום היחסי המידע מועבר למערכת הרובוטית.



תקציר

בשנים האחרונות, אחד היעדים המרכזיים בתעשייה הוא שילוב רובוטים בפסי הכללה. שילוב זה משפר את יעילות היצרנים ומפחית עלויות יצור. אתגרים רבים טמונים בשילוב רובוטים בקווי יצור, אחד המרכזיים שבהם הוא טיפול יעיל בסוגים שונים של אי ודאויות, בפרט במשימות הרכבה עדינות המשלבות אינטראקציה בין מספר גופים בתהליך. פעולות אלה דורשות ידע מקדים על המיקום היחסי בין האובייקטים המשתתפים בתהליך המאפשר דיוק בהרכבה על ידי רובוטים.

שערוך מיקום יחסי בין אובייקטים הינה משימה שנחקרת שנים רבות. מרבית השיטות שפותחו למשימה זו מתבססות על מידע מקדים על הגופים כגון מודלי תיב"מ. בעוד ששיטות אלו הניבו תוצאות טובות בשערוך המיקום היחסי בין גופים קשיחים, פחות הצליחו כאשר התמודדו עם גופים גמישים. לא רק בשל החוסר במודל תיב"מ תלת ממדי עבור גופים אלה, אלא גם בשל אי ודאויות נוספות הנכנסות למערכת בשל אופיים הגמיש.

לאחרונה, אחת השיטות היעילות ביותר לחיזוי מרחק יחסי בין אובייקטים הינה תוך שימוש במצלמות עומק, בשל כניסתן לשוק של מצלמות עומק זולות כמו Microsoft Kinect [3] ו- Intel RealSense [4] עלה השימוש במוצרים אלה. שימוש במצלמות אלו מאפשר לנצל את מידע העומק ולנתח בצורה טובה ומדויקת יותר את מרחב העבודה התלת ממדי.

כיום, נהוג להשתמש בשיטות מבוססות רשתות על מנת לזהות ולשערך את המרחק היחסי בין הגופים בזמן אמת בתהליך ההרכבה. שיטות אלו נבדלות ביניהן בזמני הריצה והדיוקים. כמו כן, קיימות שיטות המשלבות מידע עומק וצבע ברשת אלה. עם זאת, מחקרים רבים מתבססים על מסדי נתונים סינתטיים אשר אינם משקפים את הסביבה הריאליסטית של המפעל. רעשים, הסתרות, השתתפויות ואי ודאויות נוספות נכנסות למערכת כאשר נעשה מעבר לעולם האמיתי.

עבודת מחקר זו מציעה שיטה להערכת המיקום היחסי בין אובייקטים קשיחים ולא קשיחים כאחד במשימות הרכבה בזמן אמת באמצעות ראייה ממוחשבת ושיטות למידה. במחקר זה פיתחנו ויישמנו שיטה לסיווג ושערוך מיקום יחסי בין אובייקטים גמישים בתהליכי הרכבה בעולם האמיתי, באמצעות שילוב מידע עומק וצבע בשיטות למידה עמוקות. במסגרת המחקר התקיים שיתוף פעולה עם התעשייה, במטרה לבחון את ביצועי הכלי בתהליך החדרה בסביבה תעשייתית. על מנת לבחון את השיטה, הגדרנו שלוש משימות PIH שונות:

- Peg in Hole - משימה זו הינה משימה שכיחה בעולם הרובוטיקה אשר היוותה נקודת עוגן ובסיס למחקר. משימה זו כוללת החדרת גוף גלילי לחור כאשר שני הגופים קשיחים.
- חיווט חוט חשמל - במשימה זו נעשית החדרת חוט חשמל גמיש בעובי $1[\mathrm{mm}]$ למחבר קשיח. משימה זו ברמת קושי גבוהה מאשר המשימה הראשונה ומהווה בחינה ראשונית של השיטה עם אובייקטים גמישים.
- חיבור צנרת רפואית – משימה זו הינה המורכבת ביותר שבדקנו. במשעמה זו נעשה חיבור בין צנרת רפואית גמישה ומחבר גמיש. משימה זו בוחנת את הביצועים של השיטה בהחדרת אובייקטים גמישים.

המחקר בוצע בהנחייתה של פרופסור ענת פישר,
בפקולטה להנדסת מכונות.

מחבר/ת חיבור זה מצהיר/ה כי המחקר, כולל איסוף הנתונים, עיבודם והצגתם, התייחסות והשוואה למחקרים קודמים וכו', נעשה כולו בצורה ישרה, כמצופה ממחקר מדעי המבוצע לפי אמות המידה האתיות של העולם האקדמי. כמו כן, הדיווח על המחקר ותוצאותיו בחיבור זה נעשה בצורה ישרה ומלאה, לפי אותן אמות מידה.

חלק מן התוצאות בחיבור זה פורסמו כמאמרים מאת המחבר ושותפיו למחקר בכנסים ובכתבי-עת במהלך תקופת המחקר, אשר גרסאותיהם העדכניות ביותר הינן:

E. Cohen, S. Hazan, R. Schneor, A. Fischer, and M. Zacksenhouse, "Learned impedance policies facilitate robotic assembly in general direction and integration of visual sensor," Under Review, 2024.

S. Hazan, R. Schneor, and A. Fischer, "Estimation of relative position between objects for robotic insertion of LDOs using classification learning methods," in IMVC, Israel Machine Vision Conference, Poster presentation, 2024.

הכרת תודה מסורה לטכניון על מימון מחקר זה.

שערוך מיקום יחסי בין אובייקטים לשם החדרה רובוטית של גופים גמישים בשימוש שיטות קלסיפיקציה בלמידה עמוקה

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת מכונות

שר חזן

הוגש לסנט הטכניון — מכון טכנולוגי לישראל
ניסן התשפ"ד חיפה מאי 2024

שערוך מיקום יחסי בין אובייקטים לשם החדרה רובוטית של גופים גמישים בשימוש שיטות קלסיפיקציה בלמידה עמוקה

שר חזן