Machine Learning-based Control for Adaptive Human-Robot-Collaboration in Manufacturing

Ruidong Ma

Supervisors:

Dr John Oyekan
Prof. Sanja Dogramadzi
Dr James Law

A thesis submitted in partial fulfilment of the requirements
for the Degree of Doctor of Philosophy

in the

Department of Automatic Control and System Engineering

June, 2024

Acknowledgements

Thanks to my supervisors, Dr John Oyekan, Prof. Sanja Dogramadzi and Dr James Law, for their support throughout my PhD study. Special thanks to John, for his invaluable guidance and mentorship throughout my academic journey.

Thanks to my colleagues from the university for their endless help in building the experimental environments, and providing critical feedback and discussions.

Thanks to our group's industrial partners from Worcester Bosch and Satellite Applications Catapult for their vital discussion and feedback on the use-case studies. They have helped me gain valuable knowledge of the research in the industry.

Thanks to my friends in the UK and China who helped and supported me during my study. Thank you for your camaraderie, for the laughter we shared, and for all the memories we created along the way.

Thanks to my girlfriend, whose love, understanding, and companionship have been my sanctuary during the challenging times of the COVID-19 pandemic. Your resilience and unwavering support have been a source of peace and motivation. And to our cat, who has been more than a pet but a comforting presence.

Lastly, I would like to express my deepest gratitude to my father, mother and little sister. Your sacrifices, patience, and faith in me have been the driving forces behind my pursuit of this PhD. This achievement is as much yours as it is mine. Thank you for being my constant source of strength and inspiration.

Abstract

Human-Robot-Collaboration (HRC) can be seen as a promising way to meet the increasing demand for mass customization in the current manufacturing industry. It can effectively assist human co-workers and eliminate several human factors. It can also offer more flexibility and be cost-effective when compared to the industrial robot. This thesis focuses on developing machine-learning-based frameworks for collaborative robots (cobot) for HRC in industrial settings. The proposed frameworks enable collaborative robots to adapt to multiple variations during long-horizontal tasks in Make-to-Order scenarios through intuitive learning processes.

Firstly, a two-level motion planner framework based on Learning-from-Demonstration (LfD) is proposed for the robot trajectory learning problem. The proposed motion planner is able to integrate diverse trajectories for different pick and place tasks. A hybrid guidance framework combining model-based and model-free learning methods is thus proposed to improve the task success rates. The results show that it can effectively guide the learning process of a Deep Reinforcement Learning (DRL) agent by setting a proper guidance ratio.

Secondly, a novel Task and Motion planning framework is proposed for sequential HRC in massive Package-to-Order (mPTO) problems. This framework uses a Graph Neural Network (GNN)-based discrete high-level reasoning module. By effectively reducing the observation dimension through this reasoning module, the previous low-level motion planner can imitate the motion planning strategy. The results showed that the proposed framework is able to generate continuous new trajectories with unseen

tasks without the need for environmental exploration.

Thirdly, a vision-based task planning framework for responsive HRC that can work in Assemble-to-order scenarios is proposed. In this framework, a hand-centric detector is used to detect varying assembly actions without object detection. These results can be transferred into graphic observations and thus a GNN-based planer can predict the human intended goals and provide detailed assistive actions for the robot. It is generalizable to new human action sequences and thus produces new robot plans.

Finally, a framework that iteratively generates grasping solutions is proposed to further relieve the constraints in the Assemble-to-Order problem. This framework encodes the visual features of assembly parts or objects into graph observations and iteratively proposes solutions to make the human-required objects graspable while respecting the capability constraints of the robot. The results show that it can effectively generate solutions for human-required objects.

Contents

List of Figures

List of Tables

List of Abbreviations

HRC ..... Human-Robot-Collaboration

MTO ..... Make-to-Order

PTO ..... Package-to-Order

ATO ..... Assemble-to-Order

ROS ..... Robot Operating System

LfD ..... Learning from Demonstration

DRL ..... Deep Reinforcement Learning

MDP ..... Markov Decision Process

TAMP ..... Task and Motion Planning

DDPG ..... Deep Deterministic Policy Gradient

HER ..... Hindsight Experience Replay

GMM ..... Gaussian Mixture Model

HTN ..... Hierarchical Task Networks

VI ..... Variational Inference

LSTM ..... Long-Short-Term-Memory

MLP ..... Multi-layer Perceptron

CNN ..... Convolutional Neural Networks

NN ..... Neural Network

GNN ..... Graph Neural Network

GCN ..... Graph Convolutional Network

Sage ..... Graph SAmple and aggreGatE Network

RF ..... Random Forest

GPR ..... Gaussian Process Regression

SR ..... Success Rate

RRT ..... Rapidly-exploring Random Trees

LSTM-FCN .. Long-Short-Term-Memory with Fully Convolutional Network

SG ..... Semantic Guidance

SC ..... Semantic Control

VGG ..... Very Deep Convolutional Network

Faster-RCNN Faster Region-Convolutional Neural Network

WL-GNN .... Weisfeiler-Lehman Graph Neural Network

LSTM_Att ... LSTM with Attention Mechanism

Chapter 1

Introduction

The industrial sector has witnessed a series of revolutionary shifts, from the mechanization of the first Industrial Revolution to the digital transformation of Industry 4.0. To adapt to increasing market demand, manual assembly systems can be used, although this may lead to a decline in productivity due to changes in quality and fluctuating labour rates. By comparing the capability of the manual operator with the automated system, it can be seen that the performance of manual assembly is greatly affected by ergonomic factors, with limiting factors being the part weight and precision of the manual operator. The utilization of robots has been seen as a potential solution to automation. Due to its capability of carrying out repetitive tasks reliably and thus it enables the swift and efficient completion of tasks.

There are a variety of types of robots that have been used in the manufacturing sector. For instance, mobile robots, industrial robot arms and collaborative robots. Mobile robots in manufacturing are often acquired to perform mobile tasks in factory and production environments, such as material handling and warehouse management. They are often not suitable for object manipulation or assembly tasks.

Industrial robot arms are designed for heavy load-handling tasks with repetitive cycles independently. Thus, they often have large sizes and require space, when considering the safety issue. However, the flexibility and agility required for complex

assembly tasks may be too expensive to achieve [1].

Collaborative robots, also known as cobots, are a type of robotic arms specifically designed to work closely and collaboratively with human co-workers. They are often lightweight and cost-effective compared to industrial robot arms. These robots combine the capabilities of traditional robots with the flexibility and agility of human operators. The primary advantage of collaborative robots, particularly in manufacturing systems, is their ability to assist human operators rather than replace them, which means they supplement human abilities to perform tasks. Unlike traditional industrial robotics, collaborative robots offer a higher degree of safety and flexibility [2]. Additionally, they can combine the precision and speed of machines with the dexterity of human hands [1]. Collaborative robots can also easily learn from both human and programmatic demonstrations [1].

Therefore, this thesis is dedicated to the design of systems that allow collaborative robots to work automatically with humans, such systems are often referred as the Human-Robot-Collaboration (HRC) system. The collaboration states a joint activity between human and collaborative robots to achieve given tasks together. It often requires synchronous and coordinated actions from different parties while physical interaction may exist. The International Federation of Robotics [3] defined four types of HRC in the current manufacturing industry:

1. Coexistence: Human and robot work alongside each other without a shared workspace.

2. Sequential Collaboration: Human and robot are active in a shared workspace with sequential movements, which means they do not work on the same object simultaneously.

3. Cooperation: Human and robot work on the same part simultaneously.

4. Responsive Collaboration: the robot responds to human motions. This requires accurate human action understanding.

1.1 Aims and objectives

The main aim of this thesis is to develop automotive and flexible HRC systems through the current advances in machine learning technologies.

Automation transcends the traditional mechanized processes. It encompasses the use of artificial intelligence, and advanced sensor technologies to perform collaborative tasks with minimal human intervention. This not only enhances efficiency and precision but also allows industries to operate around the clock, optimizing production rates and reducing operational costs.

Flexibility, on the other hand, is about the adaptability of the HRC system. It enables industries to swiftly adapt to market demands, and customize products. This agility is crucial in today's dynamic market landscape, where consumer preferences evolve rapidly and unpredictability is the only constant.

Furthermore, this thesis specifically focuses on the Make-to-Order (MTO) scenarios. MTO is a production strategy for customized products, which usually prepares inventory in advance and performs the final production and packaging when the customer orders are placed [4]. There are variant types of MTO, such as Assemble-To-Order (ATO), Configure-To-Order (CTO), and Package-To-Order (PTO). In such problems, the main responsibility of the robot is to perform long-horizontal tasks that can manipulate different objects based on varying human intended goals or customer requests. Therefore, the robot should acquire the ability in both high-level task reasoning and low-level motion generations.

The low-level motions are the robot's fundamental capacity. It requires the robot can generate various trajectories in response to multiple positions of objects while avoiding any collision that can happen.

High-level task learning in robots refers to the process by which robots acquire knowledge and skills to perform complex tasks that involve multiple steps or require reasoning about the task's broader context. Instead of focusing on low-level control, high-level task learning focuses on understanding the overall structure and objectives

of a given task.

From the above discussion, this thesis further proposes three main research objectives that should be addressed towards building automated and flexible HRC systems:

- Easy to program: conventional approaches for robot learning often rely on hand-coded programming methods. In the aspect of trajectory learning, the robot is often trained for a specific task via either online [5] or offline training [6]. Moreover, prior knowledge or expert-defined task rules, such as previous works in [7],[8], are often needed to facilitate the high-level task planning ability of the robot. On the contrary, the scope of this thesis is to develop more user-friendly frameworks that enable more intuitive robot training.

- Personalised HRC: personalization means the robot can handle variations in the environment caused by human co-workers. In this thesis, several variations will be studied including:

  - Object position variations: this is a common variation where the robot should pick and place the different objects from varying positions.

  - Human performance variations: the collaboration strategy of human co-workers often requires a predefined workflow to avoid any confusion that may cause to robot [9]. On the contrary, this thesis aims to relieve this constraint to allow more flexible collaboration.

  - Goal configuration variations: This is a problem for MTO scenarios, where different parts can form different end-goal configurations. This variation can be caused by customer orders. This means the robot should handle different long-horizon tasks under different goal configurations.

  - Task structure variations: it requires the robot to cognitively make decisions when a pre-defined task structure is interrupted by varying human performance. In an unbounded environment, the task structure may undergo changes due to disturbances, such as a worker altering the sequence

in which required objects are placed. Therefore, the robot should possess the capability to adapt and successfully complete the task despite these variations.

- Learn beyond demonstration: Even though the current advances in deep reinforcement learning allow the robot to adapt to various tasks. It is still data-inefficient as it needs long-time exploration and can suffer from high dimensional observations. On the other hand, the major issue of Model-based learning or Learning from Demonstration (LfD) is its generalization ability when unseen tasks or distribution occurs. In this thesis, one objective is to develop data-efficient frameworks that can improve the generalization ability of the robot in both high-level reasoning and low-level motion planning with a few demonstrations.

1.2 Thesis contribution

Following the above discussion, this thesis further makes contributions to achieve the aforementioned research objectives:

- An LfD-based two-level motion planner in continuous trajectory learning. Unlike previous studies, where they either rely on environmental explorations or heavy expert knowledge, the proposed motion planner shows generalizability to unseen goals with only a few demonstrations. Furthermore, its ability to accelerate Deep Reinforcement Learning has been proven.

- An end-to-end Task and Motion planning framework in varying Package-To-Order scenarios. This proposed framework can efficiently handle diverse goal configurations with only 60% demonstrations being trained. Moreover, it also shows zero-shot generalizations on sequential HRC, where humans can randomly pack the goal objects. Its capability in practical scenarios is further assessed highlighting its strength compared to conventional machine learning algorithms.

- A vision-based task-planning framework in the Assemble-To-Order scenario in only 2D videos. This work combines a novel hand-centric motion detector with a semantic planner. Compared to the previous works, where they often heavily rely on prior knowledge or hand-coded manipulation rules, this work enables more detailed assistive plans for diverse end goals. More importantly, it is also data-efficient and able to produce new plans when unseen human action sequences are captured.

- A vision-based iterative object-grasping solution generation framework for object handling. This work is designed to relieve the constraints during responsive HRC further. Unlike the previous visual-based manipulation framework, it aims to allow the robot to directly propose grasping solutions while respecting geometrical relationships and it can be generalizable to unseen requirements and image scenes.

1.3 Thesis Overview

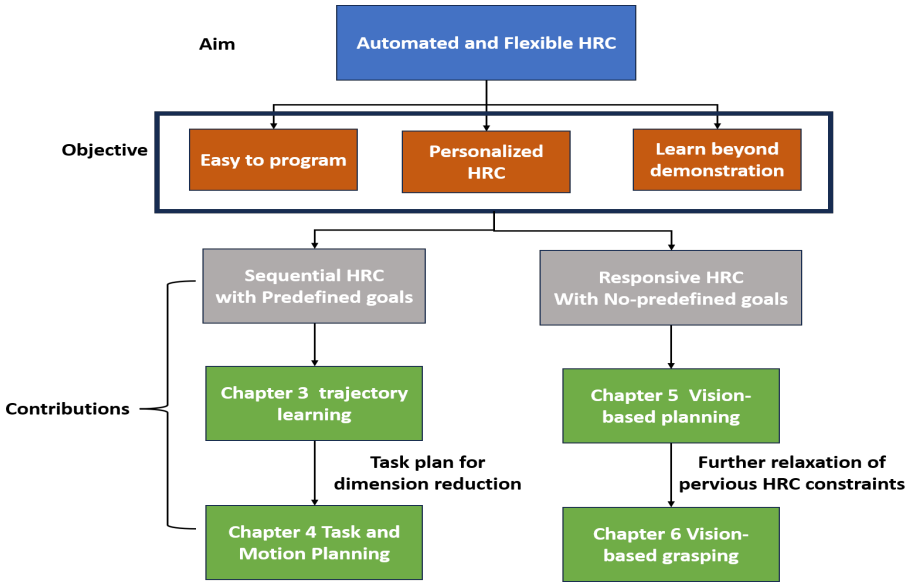


Figure 1.1: Outline of this thesis

This section describes the layout of this thesis and the works carried out in order

to achieve the aforementioned research objectives. Fig.1.1 describes the outline of this thesis.

Chapter 2 provides a literature review of the current methodologies applied to robot learning. Initially, the traditional approaches to robot learning have been explored. Following this, the chapter further reviews the state-of-the-art methodologies of model-free reinforcement learning (RL), particularly focusing on continuous trajectory learning. Afterwards, by identifying some limitations of RL, this chapter reviews the advanced model-based learning from demonstration (LfD) technologies. It mainly contains the methodology applied to trajectory learning from demonstration, high-level task learning from demonstration, and how to integrate discrete decision-making and continuous trajectory planning as a complete system for long horizontal multi-object manipulation. Finally, computer-vision-based learning for robots in the HRC field has been reviewed.

Chapter 3 introduces a two-level motion planner by imitating the expert preference and planning strategy. It further served as a semi-supervisor for DRL in a continuous trajectory learning problems.

Chapter 4 introduces a Task and Motion Planning framework that integrates discrete decision-making and continuous motion planning in Package-to-Order scenarios. The high-level decision-making module can reduce the high-dimensional observations and thus allow the two-level motion planner to adapt to diverse trajectories. It enables the robot to adapt to work solely or in sequential HRC.

Chapter 5 introduces a vision-based task planning system in Assemble-to-Order scenarios during responsive HRC. This work first built a hand-centric action detector without object detection. By decoding these actions, a planner based on graphs is able to provide detailed robot assistance actions in the form of contextual language.

Chapter 6 introduces a vision-based iterative grasping solution generation system to further relax the constraint mentioned in Chapter 5. It encodes the spatial objects in the image scene as graphic observations. By using the Graph Neural Network and Attention mechanism, it can iteratively generate grasping solutions that can make the

object required by human graspable.

Chapter 7 presents the conclusions derived from this thesis, highlighting its contributions and offering recommendations for potential future work building upon this work.

1.4 Publications

The following papers have been published and submitted during this PhD study:

- R. Ma and J. Oyekan, "Guiding Deep Reinforcement Learning by Modelling Expert's Planning Behavior," 2021 7th International Conference on Control, Automation and Robotics (ICCAR), Singapore, 2021, pp. 321-325.

- C. J. Turner, R. Ma, J. Chen and J. Oyekan, "Human in the Loop: Industry 4.0 Technologies and Scenarios for Worker Mediation of Automated Manufacturing," IEEE Access, vol. 9, pp. 103950-103966, 2021.

- R. Ma, J. Chen, and J. Oyekan, "A learning from demonstration framework for adaptive task and motion planning in varying package-to-order scenarios," Robotics and Computer-Integrated Manufacturing, vol. 82, p. 102539, 2023

- J. Chen, R. Ma, and J. Oyekan, "A deep multi-agent reinforcement learning framework for autonomous aerial navigation to grasping points on loads," Robotics and Autonomous Systems, vol. 167, p. 104489, 2023.

- R. Ma., J. Oyekan, and R. Moore. "A Learning-from-Demonstration Framework for Varying Package-to-Order Scenarios", Presented at ICRA 2023 Workshop on Communicating Robot Learning across Human-Robot Interaction

- R. Ma, J. Chen and J. Oyekan, "Graph-based semantic planning for adaptive human-robot-collaboration in assemble-to-order scenarios," 2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Busan, Korea, Republic of, 2023, pp. 2197-2203

- R. Ma, Y. Liu, E. Graf and J. Oyekan, "Applying Vision-Guided Graph Neural Networks for Adaptive Task Planning in Dynamic Human Robot Collaborative Scenarios.", submitted to Advanced Robotics.

Chapter 2

Related Work

This chapter provides a literature review that describes the fundamental concepts and current advances in machine learning-based robot control. These works form the fundamental base of this thesis.

2.1 Overview of Robot Learning

In the early stage of robot learning, there are two commonly used approaches including Lead-through and offline programming.

Lead-trough requires the teaching pendant to move the robot through the predefined trajectories. The robot will hence record the motions and produce the desired smooth motion with the safety-guarding device involved as stated in [5]. This method is simple and easy to implement. However, it is also time-consuming and difficult to implement when the programming complexity increases. Meanwhile, it needs reprogramming for every new task even if there are only a few variations between two tasks.

On the other hand, offline programming leverages computer simulations to streamline the programming process. For instance, robot coating and welding with built-in graphic simulation have been studied in [6]. In this manner, the simulated process

can be directly transferred to reality. However, it needs great programming effort and typically different robot manufacturers have different simulation platforms. Thus, the cost of acquiring one is generally expensive [2].

The conventional approaches, while effective, can encounter challenges when facing variations in tasks or workflows, particularly in the context of Human-Robot Collaboration (HRC) where different humans may have unique working strategies. This highlights the need for adaptive robot systems that can handle such variations seamlessly. As a result, there has been a growing focus on the development of automatic robot learning techniques aided by machine learning algorithms.

There are generally three classes of machine learning models that have been investigated in this thesis based on the needs, including 1. Classification, 2.Regression, and 3. Reinforcement Learning. Classification models aim to divide input data into pre-defined categorical classes, they are often used for human action or image recognition and discrete decision planning. In regression problems, there exists a continuous relationship between input data and output data, and the model's task is to learn this relationship and predict continuous outputs. They are often used for human and robot trajectory reproductions. On the other hand, Reinforcement Learning aims to produce optimal robot actions through interaction with the environment based on Markov Decision Process.

These advances aim to equip robots with the capability to learn and adapt their behaviours based on changing conditions or input from human collaborators. By leveraging machine learning, robots can improve their decision-making processes, refine their actions, and respond more flexibly to dynamic environments.

The following sections will review two main families of machine-learning-enabled robot systems. Firstly, model-free reinforcement learning will be introduced. It refers to a learning strategy in which the robot does not have prior knowledge of the tasks. Instead, it produces optimal actions by iteratively exploring the environment with rewards as feedback. In contrast to model-free methods, model-based learning or Learning from Demonstration (LfD) leverages prior knowledge typically acquired from hu-

man demonstrations. The robot learns by mapping these demonstrations into suitable actions.

2.2 Deep Reinforcement Learning

The process of reinforcement learning for robots can be modelled as a Markov Decision Process (MDP), where at each time step t, the agent selects an action a_t to maximize the obtained reward r_t given the current state s, leading to a new state s'.

In the context of a training episode, the total return $R(\tau)$ is calculated as the sum of discounted rewards $\gamma^t r_t$ from time step $t = 0$ to $t = T$, where γ is the discount factor. This factor is crucial as it ensures that future rewards are appropriately valued relative to immediate rewards, preventing issues such as infinite returns in scenarios where episodes could extend indefinitely.

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t(\tau) \tag{2.1}$$

Considering the total return of a trajectory under a policy π, denoted as $E_{\tau\sim\pi}[R(\tau)]$, involves integrating over all possible trajectories τ based on the transition probabilities $P(\tau|\pi)$ governed by that policy. This expectation helps assess the effectiveness of the policy in achieving desirable outcomes over multiple interactions with the environment.

$$\underset{\tau\sim\pi}{E}[R(\tau)] = \int_{\tau} P(\tau|\pi)R(\tau) \tag{2.2}$$

To make decisions about actions in specific states, the state-action function $Q(s, a)$ plays a significant role. This function estimates the expected return starting from a state s and taking an action a, guiding the agent's decision-making process. Under an optimal policy π^*, the optimal state-action value $Q^*(s, a)$ is maximized, leading to the selection of the optimal action $a^*(s)$ that promises the highest expected return in that state. This concept forms the basis for reinforcement learning agents to learn and

improve their decision strategies in dynamic environments like controlling a robot arm.

$$Q^*(s,a) = \underset{\tau\sim\pi^*}{E}[R(\tau)|s_0 = s, a_0 = a] \tag{2.3}$$

$$a^*(s) = \underset{a}{argmax}\, Q^*(s,a) \tag{2.4}$$

The classical learning pipeline for the RL is to store the state-action values in a matrix and iteratively update the values by active exploration of the environment. However, the problem is that it becomes impossible to select actions at given states when the state or action spaces become high-dimensional or too large, for instance, the robot manipulation problem. Therefore, approximation is necessary. The Deep Reinforcement Learning(DRL) algorithms with neural networks for function approximation has been paid a lot of attention. For the robot control problem. This approach mainly consists of two families: 1. Value-based, 2. Policy-based.

The Value-based family attempts to maximize the state-action function and find the optimal Q function. This optimization is usually performed in an off-policy manner, which means that the policy used to generate the behaviour may be independent of the policy being evaluated and improved (called the evaluation policy) [10]. There have been several approaches proposed for robot manipulation. For instance, Mnih et al.[11] implemented the Deep-Q-Network (DQN) algorithm to control a planar robot arm with 2 degrees of freedom with discrete action space. Inoue et al.[12] utilized long-short-term memory (LSTM) and a variant of recurrent neural network(RNN) to estimate the Q function in order to achieve peg-in-hole assembly. In this research, the controlled action is the force controller instead of controlling the robot directly. The limitation of the value-based method is that it can only handle the task with discrete and low-dimensional action space [13]. In contrast to the Value-based family, the Policy-based method directly optimizes policy. This method aims to maximize the parameterized policy with objective function by policy gradient. There are several approaches, for example, REINFORCE and Vanilla Policy Gradient [14].

The Actor-Critic architecture combines them together. In this architecture, it stores past exploration in the memory buffer. At the end of the iteration, the training samples are collected randomly from the buffer to train the agent. The actor selects action based on current policy and the critic for approximating the Q value to evaluate current policy. Therefore, they are combined together to iteratively maximize Q value and optimize policy by temporal difference. Trust Region Policy Optimization(TRPO) [15] is proposed to control the robot locomotion model continuously. It attempts to update policy by taking possible largest actions which is constrained by a special measurement(KL-Divergence) of probability distribution distance between the old and new policies. Consequently, they also proposed Proximal Policy Optimization(PPO) for the same control problems [16]. It is similar to TRPO while it is simpler to implement. Deep Deterministic Policy Gradient(DDPG) combines Neural Networks to solve multiple robot manipulation problems [17]. Soft Actor-Critic (SAC) was implemented for robot arm control by [18] with entropy regularization. The advantage of the Actor-Critic method is that it uses data more efficiently with a memory buffer.

The DRL approach does not acquire the knowledge of the environment at the early stage of training. In large state or action spaces, it will cause problems like dangerous exploration, and slow convergence as it explores too many unnecessary states. Hindsight Experience Replay (HER) [19]and Prioritized Experience Replay (PER) [20] are the methods to improve the sampling efficiency of the memory buffer. There are also several studies focusing on how to feed environmental knowledge to the DRL agent.

One way is to add demonstrations to DRL. Vecerik et al. [21] implemented the DDPG from demonstrations(DDPGfD). They feed the demonstration from Kinesthetic Teaching to an expert replay buffer. Thus the expert experience can be reused to accelerate learning. Similarly, Nair et al.[22] also utilized demonstrations to improve DDPG along with HER. They asked human users to demonstrate in virtual reality and filter out unsuccessful demonstrations by using Behaviour Cloning.

DRL accelerating by model-based learning has also been developed. For example, [23] utilized Guided Policy Search as a semi-supervisor to guide DDPG to learn complex

assembly tasks. Xu et al.[13] proposed a model-driven DDPG for peg-in-hole assembly tasks. In their research, they derive contact surface modelling to know the distance between the peg and the hole. Afterwards, the actor controls the feedback controller instead of directly controlling the robot arm. Du et al.[24] utilised Lyapunov barrier function to assess the safety and reachability of complicated robot control tasks through experimental data. Lin et al.[25] a Convolutional neural network(CNN) with RL agent for robotics safety navigation while considering collisions.

Moreover, the concepts of Reinforcement Learning from Human Feedback (RLHF) have drawn attention. The RLHF aims to improve the learning efficiency of RL agents by interactively obtaining human feedback or rewards on their current performance. In the robotics field, Faulkner et al.[26] proposed an interactive RL agent that can obtain rewards from both environmental exploration and human feedback. It further proposed a framework that can learn from imperfect feedback. Chen et al.[27] proposed a hierarchical framework that can efficiently map online user feedback to high level robotic tasks through noisy and stochastic inputs. Hiranaka et al.[28] leverage RLHF with primitive skill-based RL to tackle long-horizon tasks. Moreover, Zhang et al.[29] investigated offline RL with human feedback in terms of preference of trajectory pairs. They estimate the implicit reward from the offline data.

As discussed, DRL approaches are capable of handling complex control tasks with less effort in designing the control policy. However, the challenge of sample inefficiency in DRL remains a significant concern, particularly in industrial settings where training inefficiencies can lead to increased costs and potential safety risks.

Efforts in research have aimed to seek a trade-off between exploration (learning new behaviours) and exploitation (using known behaviours) to address this inefficiency. Despite these efforts, random exploration during training can still lead to dangerous configurations, posing safety risks in real-world applications.

An alternative approach gaining traction is Learning from Demonstrations (LfD) . By leveraging human expertise and guidance, this method can ensure safety during the learning process. It combines the benefits of human knowledge with machine

learning capabilities, offering a more efficient and safer path to training robotic systems, especially in complex and safety-critical environments like industrial settings.

2.3 Learning from Demonstration

The human operator can handle complicated tasks due to the flexibility of the wrist, the sensing system and the decision-making ability. Instead of directly studying the process of human learning, many researchers focus on how to map the expert demonstrations to desired robot actions which lead to Learning from Demonstration(LfD) methods. Several methods for trajectory learning only need fewer than 10 demonstrations [30]. Moreover, it can provide a way for the robot to examine the risk or security related to the demonstrated state spaces [31].

In practice, there are mainly three approaches for acquiring human demonstrations including: 1. Kinesthetic Teaching, 2.Teleportation teaching and 3. Passive Observation.

In Kinesthetic Teaching, the robot arm is directly guided by human experts and records movements via external sensors. For instance, researchers have focused on guiding robot hand to play chess [32], or polishing tasks [33]. This approach requires minimal training of the human operator and it's intuitive. Additionally, the demonstration is recorded directly from the sensory information of the robot which will simplify the following encoding and reproduction phase. However, the quality of the trajectories often depends on the quality of the demonstrations. Hence, the obtained data often requires post-processing and may also lead to sub-optimal problems. Moreover, as for complex tasks with robot hand or robot leg, the challenging demonstrations limit its capability[30].

Teleportation teaching allows the user to control the robot remotely through the user interface while the robot records data from internal sensors[30]. Unlike Kinesthetic Teaching, it does not require physical interaction between robot and human during training. There are several ways of controlling robots, for example, using hap-

tic interface [34] or virtual reality [35][36].

The third approach Passive Observation allows the robot to capture human demonstration motions passively. In[1], [37] and [38], they use stiffness data from human-human collaboration for human intention estimation and trajectory planning. Tracked human hand motion can also be used for robot path planning as studied in [36]. Moreover, some researchers only observe object trajectories for imitation, for example, Rusell et al.[39] utilize object tracking for picking and placing, Dillman et al.extract knowledge representation from the object in hand [40]. This approach is the simplest for the user.

Once the demonstrations have been collected, the next step is to extract their features and reproduce proper robot actions. Moreover, for long-horizontal tasks which they can contain multiple goals, one consideration should also be given as to how to produce a high-level task plan and corresponding low-level motions to sequentially achieve the final goal. The following sub-sections will review the current advances in trajectory learning for a single task, task planning for long-horizontal tasks as well as the integration of these two.

2.3.1 Trajectory Learning from demonstrations

Learning from Demonstration (LfD) methods provides a convenient approach, facilitating the mapping between human actions to robot actions and enabling rapid reproduction of trajectory-level behaviour. Probability-based approaches are often used to encode the trajectory from diverse demonstrations. For instance, Gaussian Mixture Model (GMM) is a probabilistic clustering model to represent sub-populations within an overall population with the assumption that the data is uniformly distributed. In the encoding phase, its purpose is to represent the states of motion from sampling points[41],[38]. The Hidden Markov Model (HMM) also describes the distribution of demonstration through a mixture of multivariate Gaussian. Unlike the GMM, it also analyses the transition probabilities between each component. Therefore, it can

describe spatial and temporal variability [42]. In LfD, HMM is extensively used in recognising the trajectories [38],[43],[37]. To reproduce the desired behaviour, methods for reconstructing the trajectory have been introduced. The main methodology used is Gaussian Mixture Regression(GMR) which provides a fast regression method with the joint probability density derived from GMM [32],[38]. However, the aforementioned approaches suffer from unseen situations and perturbations and thus lead to generalization problems [13].

To enhance variability and adaptability, the Task-Parameterized-GMM was introduced in [44]. This method involves deriving GMMs that are parameterized from various task frames. However, it necessitates an additional algorithm for identifying distinct task frames and lacks the capability to generalize to new, unencountered tasks. Study in [45] addresses these limitations by employing optimization based on reinforcement learning. Yet, the main issue of these methods is that they only concentrated on Cartesian space (3-dimensional spaces) and fall short of directly generating actions for robot joints actions.

Inverse Reinforcement Learning (IRL) is another research direction. It aims to inversely infer the potential reward of human experts while accomplishing a task. Krishnan et al.[46] proposed a framework that utilizes unsupervised learning for initial expert demonstrations and thus can improve further explorations. Zhang et al.[47] study the force-related tasks by using IRL while discovering both impedance and expert reward. However, IRL may tend to be less robust when facing imperfect demonstrations. It may not be generalizable for multi-tasks, as it needs the complete trajectory sets. On the contrary, Behaviour Cloning (BC) directly learns the policy from human demos, while it is also inefficient for complex tasks. Therefore, research often combines the BC with RL to improve the trajectory learning and generalizability [48].

Another common approach is to apply motion primitives. One common approach is Dynamic Motion Primitives (DMP), which is often used to encode the trajectory as a dynamic system based on positions, velocity, acceleration, and force.etc [43],[38]. Moreover, symbols were utilized as high-level task instructions, as demonstrated in the

work [49], while symbolic representations have been used to optimize motion primitives [50]. A study in [51] showed that a complex motion can be broken down into distinct phases, and the transitions between each phase were learned using model-based reinforcement learning. This approach depended on a pre-established library of motor primitives. Additionally, Probabilistic Context-Free Grammars were applied to construct a sequence of these motor primitives as seen in [52]. Skill trees are also applied as the segmentation of expert trajectories for long-horizon manipulation [53]. Nevertheless, these works often require a careful design of motion-primitive symbols.

An alternative approach involves leveraging hierarchical structures to break down a complex, goal-oriented trajectory into smaller sub-goals. Paul et al.identified the initial sub-goals, followed by the application of reinforcement learning for continuous control [54]. Sub-goal trees have been introduced in [55] to recursively predict positions in each task segment, while Pan et al.combined inverse reinforcement learning with demonstrations based on sub-goals [56]. However, these methods still necessitate extensive exploration of the environment, even when demonstrations are provided.

2.3.2 Task plan learning from demonstrations

Plan learning from demonstrations encompasses techniques adept at learning task-level abstractions, as discussed in [30]. Traditional planning methods typically rely on pre-established symbolic rules designed to recursively address problems specific to a domain. For instance, Kaelbling et al.[57] utilized the hierarchical nature of tasks to define task-level abstractions, enabling the formation of a planning and execution tree for dynamic planning. The study in [58] implemented Answer Set Programming (ASP) to assess the feasibility across various levels of task and motion planning.

The Planning Domain Definition Language (PDDL) represents another conventional approach. Gerevini et al.[59] developed planners focused on actions, employing symbolic descriptions and logical formulations to delineate the effects and applicability of actions. PDDL was further developed in the work [8] to facilitate temporal planning

and further expanded to address hierarchical planning challenges in [60].

Hierarchy in demonstrated task structures has been explored. Task structures can be constructed using a predefined AND/OR graph while considering all possible plan combinations [61][62]. Darvish et al.[63] developed a hierarchical architecture for HRC that can perceive human actions and therefore build representations through AND/OR graph and finally make task decisions. Similarly, DeMello et al.[64] employed an AND/OR graph to represent all potential assembly plans in tasks involving the manipulation of multiple objects. Meanwhile, the integration of this approach with DRL to accomplish flexible assembly processes for a single end product has been studied in [61] The Hidden Markov Model (HMM) is also used to figure out hidden parts of sub-tasks, as shown in [65]. Hierarchical Task Networks (HTN) are used to set up task structures and calculate how likely different steps are to happen, as seen in [66] and [67]. HTN was also used in [49], where they mixed symbolic and geometric planning to plan tasks and movements. This needed special rules and knowledge from experts to understand the effects of the symbolic planner. These methods still need careful planning based on expert knowledge. The Markov Decision Process (MDP) was used in [4] to manage production and inventory in Make-to-Order (MTO) problems. Yu et al.[68] also looked at MTO problems, using group-based scheduling with the tabu search algorithm.

The methods mentioned earlier usually don't require a training process. But, most of them need manually written task descriptions, models for how tasks change (like task structure), or rules, such as if-else statements to decide what actions are possible. Because of this, these methods often need experts in the field to create the rules while considering all the different situations that could happen.

Recent progress in Learning from Demonstrations (LfD) has opened up possibilities for automatically learning tasks without needing any hand-coding process. Pirk et al.[69] managed to abstract tasks at a high level using natural language to describe action symbols. These symbols were derived from sub-tasks shown in video demonstrations using a Sequence-to-Sequence model. However, it's not clear how they com-

bined these action symbols with motion plans based on position. Inverse reinforcement learning (IRL) was applied to determine the best actions that yield optimal rewards, specifically for customized products as shown in [9].

Researchers are increasingly using Graph Neural Networks (GNN) [70] for task-level abstraction, thanks to their ability to naturally represent relationships between objects or multiple tasks. Some research focuses on turning symbolic task descriptions into graph formats. For instance, Hayes et al.[66] used a Conjugate Task Graph (CTG) to create sequences of sub-tasks, while Su et al.[71] developed manipulation graphs that include sets of motor primitives for performing manipulation tasks. Huang et al.[72] introduced a Neural Task Graph (NTG), using LSTM to turn demonstrations into task nodes, creating action transitions as edges that connect valid task nodes through CTG. Li et al.[73] worked on training GNN with reinforcement learning (RL) for handling multiple objects, but their results seem less effective compared to training GNN directly with demonstrations as shown in the study [74]. Their approach shows adaptability to new tasks. However, these methods still need a lot of work to set up the ground truths, like deciding the order of sub-tasks and how each node is connected in the graph.

On the other hand, some studies have used Graph Neural Networks (GNN) to understand the relationships between objects. To create high-level abstractions, Ye et al.[75] used graphical interaction networks, while Battaglia et al.[76] captured how objects interact with each other and with robots. They then used these relationships as states in a Model Predictive Controller. Additionally, Silver et al.[77] focused on object properties rather than their positions. They used these graphic observations to rank the importance of objects and plan incrementally among many objects at once. However, their method isn't well-suited for problems that require planning in a sequence. Lin et al.[74] demonstrated that GNNs can learn task structures directly from just the information about objects, without needing predefined symbolic task descriptions with fully connected graph observations. Following this, Felice et al.[78] applied GNN to identify important objects as a form of node classification. Their work also showed the ability to generalize to new pick-and-place tasks without prior training (zero-shot

generalization). However, in their approach, they only consider the initial state of each sub-task. Once the objects and their goal positions are known, they rely on traditional motion planners for the actual movement generation.

Moreover, current advances in Large Language Models (LLM) have shown great potential in robotics, especially for vision-based task planning. For instance, Hori et al. [79] utilized an LLM-based question-answering agent that can improve robotic motion planning. Consequently, Ren et al.[80] proposed a framework that measures and aligns the uncertainty of LLM -based planer and therefore can ask for human feedback. Zhou et al.[81] proposed a framework that can iteratively improve the robotics performance which is originally proposed by LLM for long-horizon tasks. LLM shows promising abilities in commonsense reasoning and planning. However, this method needs large-scale computation and training datasets.

2.3.3 Task and Motion Planning

Efforts have been dedicated to integrating the aforementioned discrete plan planning and continuous trajectory learning methods mentioned earlier into a unified framework. These methods are commonly referred to as Integrated Task and Motion Planning (TAMP) [82].

The main challenge in Task and Motion Planning (TAMP) architecture is combining a discrete task planner with a continuous motion planner. One common way to solve this is through sampling-based methodologies. These methods blend discrete task planning and continuous motion into a single search space. They use sampling-based probabilistic searches to explore and find solutions within this space [83]. Fang et al.[84] sampled valid sub-goals and actions using cascaded Variational Inference and a user-defined reward function. Another method, used in [85], involves applying a conditional sampler with domain knowledge to pick actions in a large solution space. However, these sampling-based methods can be inefficient in Make-to-Order (MTO) scenarios, where the task-level search can become complex. Additionally, if the hierarchical nature

of trajectories is taken into account, these methods might not find any solutions at all [83].

To address the limitations of sampling-based methods in Task and Motion Planning (TAMP), one possible solution is the use of Procedural Attachment [83]. This approach involves a high-level task planner followed by an external motion planner [83]. Several studies have applied Procedural Attachment in flexible production with robots. For example, Kurosu et al.[86] used Mixed Linear Programming (MILP) to prioritize the sequence of objects, and Behrens et al.[87] employed Ordered Visiting Constraints (OVC) with constraint optimization. However, these techniques require a pre-known symbolic representation of the task structure to manipulate workpieces sequentially towards a specific goal configuration. This need for a predefined task description means they struggle with environmental variations. Therefore, if a customer order changes, it becomes necessary to redefine the task description and retrain the model. Additionally, in these implementations, only high-level plans are considered, while low-level, collision-free motion generation is left to existing motion planners like Rapidly Exploring Random Tree (RRT), which can be time-consuming.

To handle variations in observed objects, recent advancements in Graph Neural Networks (GNN) [74] have demonstrated the capability to use graph-encoded representations for learning high-level policies, rather than relying on symbolic representations. For instance, Lin et al.[74] showed that this approach can learn task-specific rules and adapt to varying geometric goal configurations by utilizing objects' observations during demonstrations. However, their method only addresses scenarios where the task structure remains unchanged. This means it's not equipped to handle redundant objects within a task structure or across different tasks. Additionally, their approach doesn't accommodate variations in the sequences of tasks involving objects, which is crucial for the human element in Human-Robot Collaboration (HRC) scenarios.

As discussed, LfD approaches often encounter generalization issues. The main reason is that demonstrations are often constrained to specific expert policies. Since the environment may not be fully searched by demonstration, the agent is likely to fail

when the unseen input arrives [30]. Efforts have been made to improve the robot's generalization ability by combining diverse potential manipulation rules as seen in AND/OR graph or Hierarchical Task Networks. However, these solutions need careful design with domain expert knowledge.

2.4 Visual manipulation for Robotics in HRC

This thesis also focuses on how to interpret the demonstrated visual human information as well as objects and thus produce corresponding robot actions. Therefore, this section provides a literature review of vision-based human and object understanding.

2.4.1 Human action understanding

Human action understanding is the key factor of HRC in terms of safety measurement and robot programming. By understanding human actions, the robot can infer human intention and therefore cognitively make decisions or actions that can assist humans further. Unlike the traditional user interface like a keyboard and touchscreen, vision information can relieve the burden of communications in HRC. As a result, it leads to intuitive programming which offers an easy robot programming approach for humans with non-expert knowledge.

For different assembly actions, human gestures can be unique. As the human body and hands have unique features, for instance, skin colour and body shape, compared with environment [88]. Some of the research has focused on identifying human gestures through spatial images. For instance, Brethes et al.[89] proposed a framework to automatically segment the picture and detect skin colour for hand and body recognition. The drawback of this method is that it can be affected by environmental illumination and human races. The local features approach has been proposed to eliminate the illumination effect, which is less sensitive to lighting. It is a detailed texture-based approach which segments the image into small parts and does not correspond to body

parts according to Speeded up robust features (SURF) by [90] and Oriented FAST and rotated BRIEF (ORB) by [91]. However, only analysis of human gestures can be inefficient as human gestures may vary from time to time, even for the same actions. As a result, current advances focused on utilizing Neural Network (NN) approaches to enable more robust and efficient scene recognition. Chen et al.[92] adopted convolutional neural networks (CNN) to perform scene understanding including the information about human and environmental surroundings to infer human actions. Zhang et al.[47] proposed a more detailed dual-CNN model, where one CNN identify the human pose while the other considers the objects' context.

In the HRC environment, human motions are also crucial to enable the robot to recognise the assembly intention. In order to identify and correctly track the movable human in different frames, several gesture-tracking solutions have been proposed. One solution is to represent the human in only one hypothesis which refers to single hypothesis tracking, for instance, Mean shift tracker[93] and Kalman Filter[94],[95]. To classify the tracked motions, it often relies on machine learning algorithms. K-nearest neighbour (KNN) is able to classify user movements using the closest training points. Given that the observation is a continuous sequence, the Hidden Markov Model (HMM) and Gaussian Mixture Model (GMM) are able to extract the desired states. Thus, the test data sets can be classified by comparing extracted states with the trained model [96]. Current works also focus on utilizing Neural Network (NN)-based algorithms for classification, for instance, Recurrent Neural Networks [7] as well as semi-adaptable Neural Networks [97] with skeleton data or hands.

Combining temporal motion features with spatial assembly context can be more effective in distinguishing various assembly actions, especially in Assemble-to-Order problems [98]. When creating a variety of products, different components may be placed in the same position based on customer demand. In this scenario, the assembly motions, such as moving to a particular position, maybe the same, but the corresponding objects can vary. Many existing approaches often employ Convolutional Neural Networks (CNN) for object detection and Recurrent Neural Networks (RNN) to recog-

nize assembly actions, utilizing data from depth image sequences [99] or skeleton data [61].

For long-horizon sequential collaborative tasks, the robot should thus recognise the human intended the end-goal and plan based on observed human action sequences. Cognitive architectures have been studied for this purpose, for instance, Learning Intelligent Distribution Agent (LIDA) [100], Adaptive Control of Thought (ACT) [101] and ACT-Rational [102],[103]. they can integrate the low-level perception knowledge and extend to necessary control behaviour, for instance, motion planning, decision making, and memory[104]. However, there are still existing problems, for example, high dimensional data can cause the decision to become unstable [104]. Probabilistic approaches have found extensive applications in estimating human intended goals. For example, Bayesian Inference has been employed to infer human navigation goals [105] and to determine assembly plans using prior knowledge of human pose and object interaction as studied in [7]. Additionally, Variable-Length Markov Models (VMM) have been utilized to analyze classified action sequences, leading to the generation of optimal plan predictions [47]. It's worth noting that these approaches often rely on prior knowledge provided by domain experts.

2.4.2 Robot manipulation

Computer vision also plays an important role in robotics manipulation tasks. It enables the robot the ability to sense the environment and provide feasible solutions for grasping the desired objects even in multi-object scenarios.

Towards addressing this problem, one research direction is to model the geometric property of the target object. Generative Grasping Convolutional Neural Network (GG-CNN) has been proposed in [106] and [107] that is able to predict the grasping quality and grasping pose for the end-effector based on the pixels from the object's depth image. Sundermeyer et al.[108] proposed an end-to-end grasping network that can directly generate a distribution of 6-DoF parallel-jaw grasps. By determining the

full 6-DoF grasp pose and width in the observed point cloud, the dimensionality of the grasp representation can be effectively reduced to 4-DoF. Zhang et al.[109] analysed the point cloud features based on the estimated 6D pose from Pixel-Wise Voting Network (PV-net). Huang et al.[110] equipped an event-based camera system to generate a grasping strategy with the point cloud information. Unlike the aforementioned studies which often focus on grasping pose estimation, Wang et al.[111] also concerns the generation of the collision-free trajectory generation. They proposed a hierarchical framework that learns goal-driven grasps based on partial point cloud observations. During training, an embedding space is learned to encode expert grasp plans, and during testing, a variational autoencoder is used to sample diverse grasp trajectories. The aforementioned methods often require rich information about the target object, for example, point cloud, or CAD model. Meanwhile, it can be problematic when the target object is overlapped by the surrounding objects too much and therefore the direct grasping pose estimation may not be effective.

One possible solution is to reason the objects' relationship through the visual feature, and thus allow the robot to grasp the intended object orderly. For instance, Li et al.[112] consider the objects' pair and proposed visual phrase-guided CNN. Qi et al.[113] constructed spatial and temporal object graphs through detected features and thus detected the human-object interaction through a proposed Graph Parsing Neural Network (GPNN). Lu et al.[114] combine the detected spatial, semantic and visual information from object pairs and produce the object interaction status through CNN and tree-based gradient boosting models (GBMS). Consequently, Lu et al.[115] studied the same problem by using Sequence-to-Sequence (Seq2Seq) with the Transformer model. Guermal et al.[116] proposed a three-step framework which can reason the object-object relationship in a temporal video clip. For the specific application in robotics, Goodwin et al.[117] investigated the rearrangement of unseen but similar objects with seen goal images by leveraging the semantic and visual information through Contrastive Language–Image Pre-training (CLIP). Huang et al.[110] study the objects' relationship over the robot manipulation via partial point cloud and Graph Neural Net-

work. Ardon et al.[118] constructed a knowledge graph representation using Markov Logic Networks to obtain the probability distribution of an object's grasp availability. Moreover, there is also attention been paid to the specific object stacking environment using CNN as in [119],[120]. This means that CNN would need to systematically examine every possible relationship between each pair of objects in the image. GNN has also been applied to predict the relationship nodes through the encoded graph observation [121],[122].

2.5 Summary

This chapter has introduced the concepts of traditional robot learning approaches, reinforcement learning and learning from demonstrations specifically for trajectory generations and task plan learning, and computer vision technologies on human action understanding and robot manipulations. They form the foundations of the thesis, and they can be concluded as follows:

- Section 2.1 initially provide the traditional methodologies in robot learning. As stated, they lack adaptiveness in regard to environmental changes.

- Section 2.2 provides a general background of reinforcement learning for the work in Chapter 3. It is a model-free learning scheme that allows the robot to explore the environment while building the state-action transitions through Markov Decision Process. Current advances in DRL have shown great potential in tackling complex control problems. However, sample inefficiency is a major issue, especially in industrial settings. Therefore, Chapter 3 proposes a novel LfD framework that can effectively guide the exploration and the importance of diversity in memory buffer will be studied.

- Section 2.3 review another robot learning method as Learning from Demonstration (LfD) in both trajectory reproduction and task planning. For robot trajectories, probabilistic models are often used to model diverse demonstrations for

one end goal. These studies only focused on the 3D positions of the end-effector. Motion primitives are also used to encode the trajectory as a dynamic system with primitives, such as symbolic representations. Moreover, hierarchical structures can be used to decompose the more complex goal-oriented trajectories into subgoals and actions, while the current studies still need extra exploration to guarantee generalization ability. Therefore, Chapter 3 proposes and studies a novel motion planner that can handle diverse expert trajectories efficiently and produce direct joint actions. It holds the generalizability without the need for exploration.

Task planning, on the other hand, plays an important role in long horizontal tasks. It aims to learn the high-level task abstractions for each sub-task. Traditional methodologies often use predefined symbolic task abstractions while it needs hand-coded manipulation rules. Therefore, Chapter 4 aims to utilize machine learning methodology (specifically Graph Neural Networks) to automatically learn the underly task strictures from the demonstrations without hand-coding.

- Section 2.3.3 introduces TAMP architecture that can integrate the aforementioned continuous motion generation and discrete task planning. Sampling-based methodologies merge the task and motion planning into one search space and find the solution through probabilistic search, while it is not able to handle complex tasks. Procedural attachment is another direction that produces actions in correspondence with a high-level planner. Based on this, Chapter 4 further proposes a TAMP system by combining the proposed task and motion planner following Procedural attachment methodology.

- Section 2.4.1 reviews the vision-based human action understanding for the part of the work in Chapter 5. Human gestures can be a vital source for different assembly actions. It can be detected through skin colour classification, while NN can be more robust when considering object context in the scene. Human motion

predictions are also important. Current approaches often utilize probabilistic models and NN-based methods. Chapter 5 considers combining the temporal motion features and spatial assembly context, which has been proven to be more effective in distinguishing different assembly actions, especially in MTO scenarios.

Afterwards, the robot should recognise intentions and plan through the observed action sequences in long-horizontal MTO tasks. Efforts have been made to develop cognitive architectures by integrating perception knowledge and control behaviours, while they may suffer from high dimensional observations. Another research direction is to utilize probabilistic approaches, which are often heavily based on prior expert knowledge. Thus, Chapter 5 further proposes a novel task planner that can process graphic human action sequences and hold generalization ability in unseen observations.

- Section 2.4.2 offers a review of vision-based robotics manipulation approaches for Chapter 6. For a scene containing multiple objects, some of the current research focuses on estimating the optimal grasping pose. These approaches often need rich object information. Another research direction is to reveal the visual relationship between the objects and thus build manipulation trees based on classified relationships. Based on these, Chapter 6 proposed a cost-effective visual solution generation system that aims to produce grasping actions directly from image scenes.

Chapter 3

Continuous trajectory learning

3.1 Introduction

This section aims to tackle the motion planning problem of robots in multi-object environments while dealing with variations including object positions and manipulation order variations. Specifically, this chapter studies sequential HRC for adaptive human error correction.

The presence of humans in manufacturing may introduce disturbances into the system and result in variations and unpredictability. As a result, product quality could be highly impacted [123]. In assembly processes, common errors include omitting items, the application of wrong components and improper installation to mention a few [123]. Current approaches in error handling are limited to classifying human errors [123], allowing humans to override robots [124] or focus on only specific types of errors, for instance, positioning [125].

In situations where there are various objects to pick and place, learning just a single trajectory isn't adequate. Collecting a range of demonstrated trajectories can be time-consuming and resource-intensive. At the same time, the safety of these objects during manipulation is a crucial factor that needs to be taken into account.

To tackle these challenges, this work decomposes the motion planning problem into

hierarchical two-level steps including the following components:

- Subgoal planner is a variational-inference-based probabilistic regressor. By accessing information about a task, such as its label, the subgoal planner can effectively integrate different trajectories for various picking and placing positions. This integration is crucial for handling tasks that involve moving objects to different locations, as it allows the planner to adapt to a variety of scenarios by predicting the necessary subgoals based on the specific task at hand.

- For executing the actions related to each sub-goal through final joint actions, an action planner is developed. This planner is capable of directly learning actions by modelling expert preferences using a simple neural network. This approach differs from [126], where neural networks were employed to learn the state transition function and, in turn, guide the action learning of Deep Reinforcement Learning (DRL). The method presented in this study enhances data efficiency and eliminates the necessity for extensive exploration.

However, it has been noted that the subgoal generation could suffer from high dimensional observation. To improve the success rate, this Chapter further proposes a hybrid hierarchical guidance architecture in order to guide DRL agents for more efficient learning.

Consequently, there are serval research questions that this Chapter aims to address:

- **Q1:** Is it efficient enough to only model the function of expert preference (i.e. goal-state positional difference) for dealing with variations in object positions?

- **Q2:** Can it produce collision-free subgoals through raw observations while dealing with objects' position variations and manipulation order variations?

- **Q3:** There will also be experiments that are designed to examine the most appropriate way of guiding DRL with demonstrations.

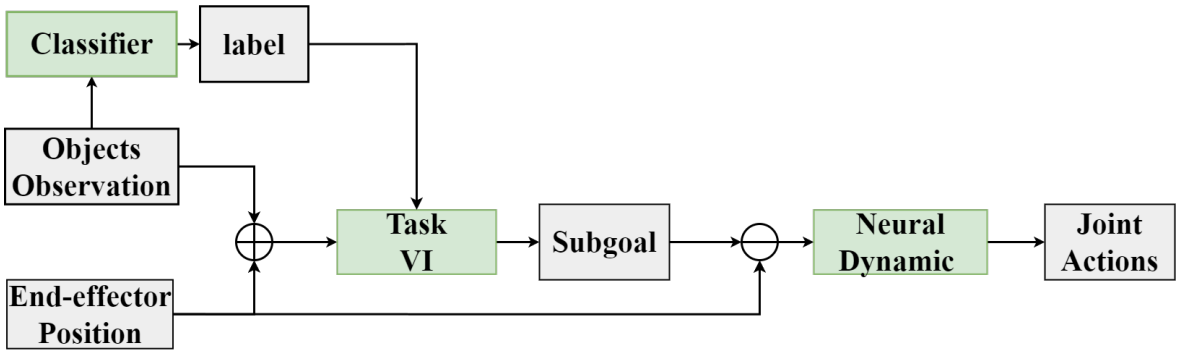


Figure 3.1: two-level motion planner

3.2 Methodology

Fig.3.1 shows the proposed motion planner, the inputs of the Task-conditioned subgoal planner (TASK VI) are full observations of objects and end-effector positions. A simple classifier is used to identify the task stages as **picking** or **placing**. By modelling the expert preference as goal-state (or subgoal-state) positional difference, the Neural Dynamic Planner is proposed to produce joint actions.

Moreover, as found in the experimental section, the high-dimensional observations can result in inefficient subgoal planning. Thus, this work further proposes a framework that enables more efficient trajectory learning of DRL by leveraging the proposed model-based two-level motion planner. At the early stage of training in DRL, one of the most common issues is that there will be slow convergence as it may explore too many unnecessary states. Therefore, the two-level motion planner is used as a semi-supervisor of DRL and thus guides the DRL agent to narrow down the search space.

3.2.1 Task Conditioned Subgoal Planner

In sub-goal planning, when the same object is picked for different locations, similar observations might confuse the planner. To avoid this, task parameters, such as labels, are provided to help differentiate between different tasks. Let $x \in X$ stand for the raw observations and $s \in S$ be the corresponding 3D trajectory level subgoal from

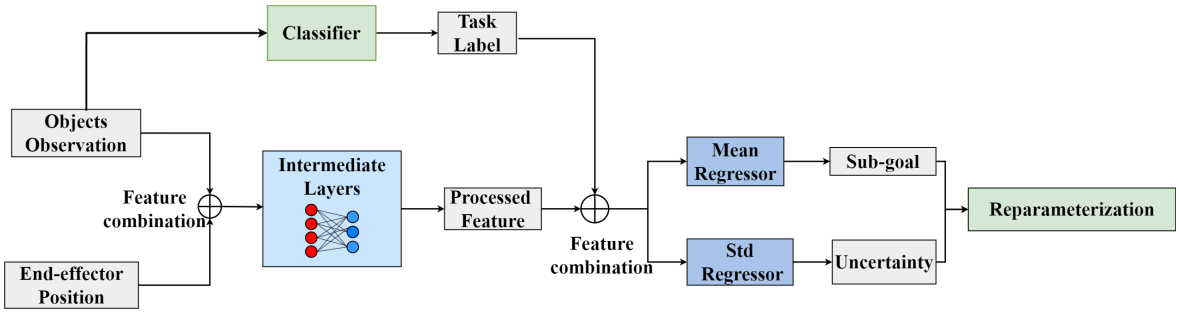


Figure 3.2: The architecture for the Task-conditioned sub-goal planner starts with an input layer that takes in raw observations from the environment. It then processes these observations through two fully connected intermediate layers to extract relevant features. These features are combined with task labels as inputs. It adopts a Variational inference to predict sub-goals, employing a reparameterization process for effective training. Finally, the Tanh activation function is used throughout the network for non-linear processing.

demonstrations. The robot end-effector is r. A classifier based on MLP is firstly built as $l = c(x; \theta_c)$, where $l \in L$ is the task label and θ_c are the training parameters of the classifier. The full observations can thus be expressed as $\mathcal{O} = \{x, l, r\}$.

Moreover, instead of using a deterministic model that directly generates the categorical distribution $p(s \mid \mathcal{O})$, a variational inference-based probabilistic regressor with additional uncertainty output δ (e.g. standard deviation(std)) is adopted. The latent parameters $Q(z) \sim \mathcal{N}(\mu, \delta)$ are formulated to approximate the ground truth sub-goal s as $p(z \mid \mathcal{O})$. The μ and δ can be parameterized with neural network as dependency of $\mathcal{O}$, $\mu = f_{\theta_{sub_\mu}}(\mathcal{O})$, $\delta = f_{\theta_{sub_\delta}}(\mathcal{O})$. According to Bayes rule, the posterior $p(z \mid \mathcal{O})$ can be expressed as Eq.3.1. The integral form of marginal likelihood $\int p(\mathcal{O} \mid z)p(z)dz$ is often computationally intractable. In variational inference, it tries to find the optimal distribution $p^*(z)$ that approximates the posterior distribution. It is equivalent to performing optimization by maximizing the Evidence Lower Bound function (ELBO) in Eq.3.2, where $E_{z\sim Q}[logp(s \mid z)]$ is the likelihood term and D_{kl} is the Kullback-Leibler (KL) divergence that regulates the predicted variational probability $q(z)$ with a prior distribution $p(z)$. The KL divergence can be rewritten as the expectation form of z and finally, the cost function can be expressed as Eq.3.3.

Assuming the likelihood and variational distribution are Gaussian according to [127], they can thus be replaced with a negative Gaussian log-likelihood function in Eq.3.4. For prior probability, the ground truth sub-goals are assumed as a unit Gaussian distribution $p(s) \sim \mathcal{N}(0, 1)$.

Fig. 3.2 illustrates the network architecture, it uses a Stochastic Gradient Variational Bayes(SVGB) estimator with reparametrization trick to train the model [127]. By accessing task labels, the sub-goal planner is able to produce adaptive sub-goals among different tasks. The variational inference concepts can lead to high likelihood while penalizing over-fitting when estimated $q(z)$ is far away from the true prior $p(z)$.

$$p(z \mid \mathcal{O}) = p(z)\frac{p(\mathcal{O} \mid z)}{\int p(\mathcal{O} \mid z)p(z)dz} \tag{3.1}$$

$$\arg\max \mathcal{O}_z = E_{z\sim Q}[\log p(s \mid z)] - D_{kl}[q(z)||p(z)] \tag{3.2}$$

$$\mathcal{L}_{sub} = E_{z\sim Q}[\log p(s \mid z)] - E_{z\sim Q}[\log q(z) - \log p(z)] \tag{3.3}$$

$$\mathcal{L}_{Gaussian} = -\frac{N}{2}(2\pi\sigma^2_{\theta_\sigma}) - \frac{1}{2\sigma^2_{\theta_h}}\sum_{i=1}^{N}(p'_i - \mu_{\theta_\mu}) \tag{3.4}$$

3.2.2 Neural Dynamic Planner for joint actions

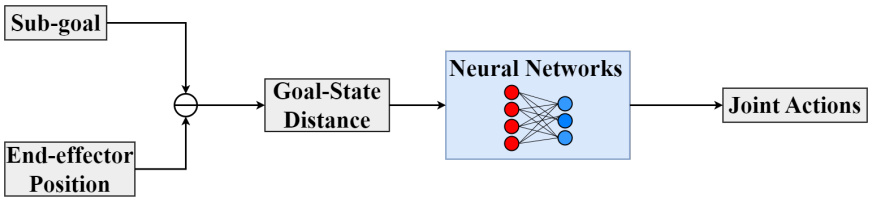


Figure 3.3: The neural dynamic planner's architecture. Its inputs consist of the distance between the predicted sub-goal and the current position of the end-effector. These inputs are then processed through a feedforward three-layer neural network. The network's purpose is to ultimately generate the joint actions needed to achieve the sub-goal. Between each layer of the network, the ReLU activation function is used.

For the action planer, since there will be a large number of observations and continuous actions, a small number of demonstrations is inefficient to directly map the observations to optimal actions required to achieve the sub-goal.

In the natural way of a human reaching task, the distance often contributes to direction preference [128]. Inspired by this, the expert preference is modelled as a dynamic transition function. Assume experts always prefer to minimize the distance between current end-effector position r_t and sub-goal position s_t at every time step t: $\Delta_t = s_t - r_t$ in a consistent way (i.e they will first minimize Δ_t in the x-y plane, and thus approach the final goal vertically). Meanwhile, the joint actions $a_t = [a_1, a_2, a_3, ..., a_n]$ will lead to different position r_t through forward kinematic in robot arm. Thus, a dynamic transition function is formulated which contains a continuous state space as Δ_t with action space $a_t = [a_1, a_2, a_3, ..., a_n]$. It aims to obtain the actions from the expert preference $\Delta_t = s_t - r_t$ inversely through a simple neural network $a_t = D_{\theta_D}(\Delta_t)$ as shown in Fig.3.3. This is equivalent to approximating the inverse kinematics of expert demonstrations. It is trained with a supervised loss function as Mean Square Error(MSE), which minimizes the loss between ground truth action a_t and prediction as shown in Eq.3.5, where T is the training batch size.

$$\mathcal{L}_{act} = \frac{1}{T} \sum_{(a_t, \Delta_t) \in T} \frac{1}{2} ||a_t - D_{\theta_D}(\Delta_t)||^2 \tag{3.5}$$

3.2.3 A hybrid guidance framework for Deep Reinforcement Learning

Inspired by [129], this work aims to integrate the expert planner with a Reinforcement Learning(RL) agent to improve success rates with only sparse reward. In this part, the problem can be formulated as a Markov Decision Process (MDP) with continuous state space $\boldsymbol{S}$ and action space $\boldsymbol{A}$. Since the proposed planner is acting on the joint action level, it is able to provide the subgoal and thus useful actions and corresponding state to allow more efficient exploration of the RL agent.

Algorithm 1 Hybrid Hierarchical Guided RL

Initialize RL agent and Experience Replay Buffer R

Trained Hierarchical Planner $\boldsymbol{h}$,$\boldsymbol{d}$,$\boldsymbol{c}$

Guidance ratio λ

for *Epoch=1,M* **do**

  if λ **then**

    Reset environment and receives s_1 and x_1

    Task Label $l = c(x_1)$

    Corresponding Final Goal position p_{goal}

    for T*=1,*H_{hi} **do**

      Subgoal $p'_T = h(x_T, l_T = c(x_T))$

      for t*=1,*H_{lo} **do**

        Select action a_t according to current policy

        Calculate future end-effector position p_{t+1} according to a_t

        Future Distance $\Delta_{t+1} = p'_T - p_{t+1}$

        if $\Delta_{t+1} > \delta_g$ **then**

          Current Distance $\Delta_t = p'_T - p_t$

          Corrected action $a_t = d(\Delta_t)$

        Execute a_t ,and receives state s_{t+1}

      Store expert transition

  else

    for n*=1,*$H_{hi} \times H_{lo}$ **do**

      Pure RL exploration

    Stores transition

  Sample a mini-batch from R

  Update RL agent

Algorithm 1 illustrates the proposed pipeline. Note that observation x fed into $\boldsymbol{h}$ and $\boldsymbol{c}$ is different from the state s for the RL agent. Deep Deterministic Policy Gradient (DDPG) with Hindsight Experience Replay (HER) is adopted as the baseline RL agent. It is tricky for an RL agent to achieve a high-dimensional end goal with only sparse rewards. Hence, at the beginning of each task, the proposed hybrid framework

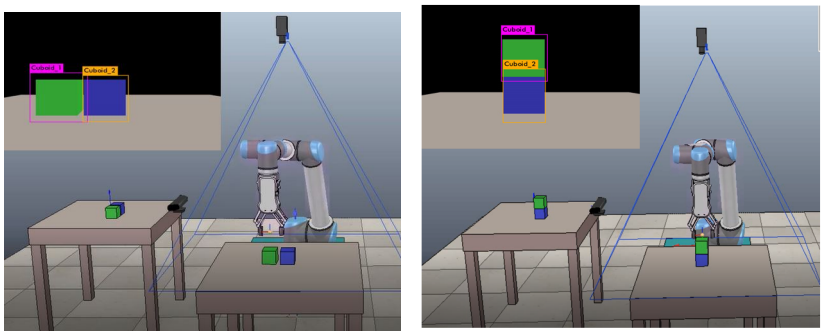
first classifies tasks $\boldsymbol{l} = \boldsymbol{C}(\boldsymbol{x_1})$ and selects the corresponding object positions. Each task can therefore be decomposed into pick-and-place tasks. Also, a critical issue with pure demonstration-guided RL agents is noted: with only informative experiences, the agent is likely to lose its knowledge as the experiences are too limited [130]. To address this problem, there are two solutions being designed: **1.** Setting up a guidance ratio λ to allow the agent to be guided or explore by itself alternatively; **2.** At every action planning horizon H_{lo}, instead of directly using expert actions, the proposed framework intend to correct the action from the agent's policy. This means that it first calculates the future distance Δ_{t+1} with forward kinematic and only correct the actions which cause Δ_{t+1} greater than a guidance threshold δ_g, which is 0.05m in this study.

3.3 Experimental setup

The experiments are built and conducted in a simulation platform CoppeliaSim [131]. In the simulation environment, a Universal 10 (UR10) robot arm with a suction cup is adapted to interact with cubic objects, where the objects are placed on a table in front of the robot. The robot is controlled by the proposed models written in Python.

The first simple experiment is to place or stack the objects based on pre-defined goals as shown in Fig.3.4. The tasks require the robot to produce actions that can move an object to the various target locations starting from a fixed initial state. Assuming the demonstrated actions can be imperfect, this experiment is designed for the Neural Dynamic Planner with DRL to assist its ability to improve DRL learning efficiency with only direction preference.

Moreover, to validate the two-level motion planner, two multi-object correction experiments are designed and simulated. The experiments are set up under the assumption that the robot should correct errors according to end goals as Fig.3.5. The human error can be reflected in different cases including the objects' placing orders (wrong components) and objects' positions (improper placing). The robot is thus involved in sequential collaboration to correct them with the green cuboid as the benchmark.



(a) Stacking. (b) Placing.

Figure 3.4: Designed tasks, the left top pictures illustrate the final goal.

Fig.3.6 illustrates the first experiment, the wrong component can be either red or blue. Fig.3.7 describes the second two objects correction experiment where objects should be corrected sequentially as objects may occupy other positions. Lastly, collisions among objects are prohibited.

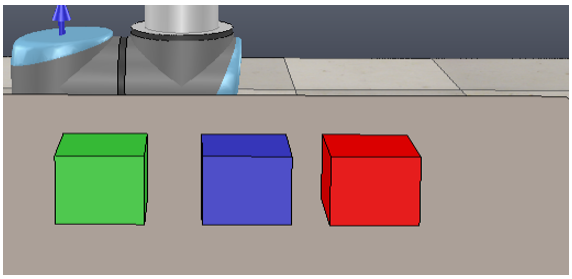
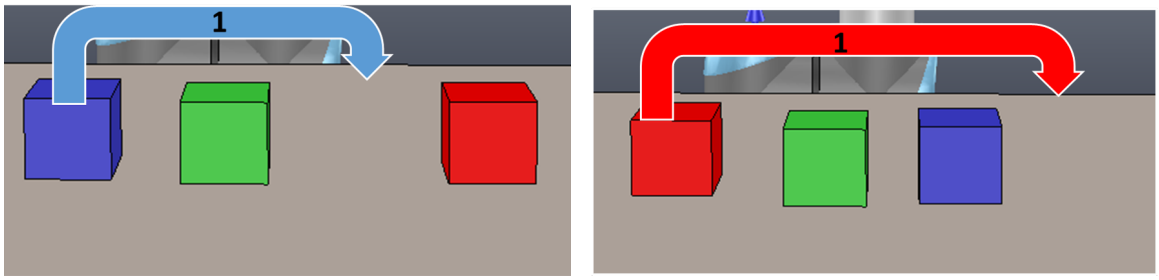


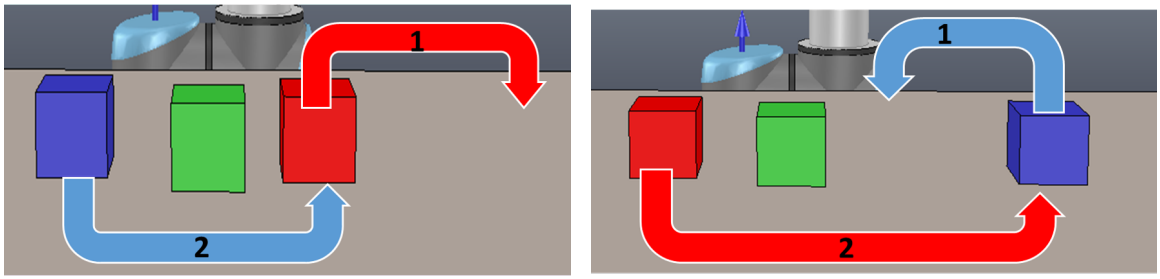
Figure 3.5: Objects placing end goal. The robot is required to place each cuboid in correct position and order (e.g green, blue and red)

For the first placing and stacking experiments, the demonstrated joint actions were collected through keyboard teleportation, which means the demonstrator controls each joint movement through keyboard command. There were 3 controlled joints including $[q'_{base}, q'_{shoulder}, q'_{elbow}]$. which affect the end-effector positions mostly. As the demonstrated actions may be imperfect, only the action direction is recorded. To judge whether a demonstration is successful, the final goal-state positional difference is measured if it is within a threshold $|\delta| < 0.15m$ as shown in Fig.3.8. Therefore, instead of producing actual actions, the Neural Dynamic Planner only generates directions



(a) Case 1 (b) Case 2

Figure 3.6: One object correction experiment with the same end goal under two scenarios. Numbered arrows represent the correction sequence



(a) Case 1 (b) Case 2

Figure 3.7: Two objects correction experiment with the same end goal under two scenarios. Numbered arrows represent the correction sequence

$d_i \in [-1, 0, 1]$ which stand for -1: Moving Negatively; 0: Not Moving; and 1: Moving Positively. The Neural Dynamic Planner can be involved anytime when the actions proposed by DRL may cause the calculated future goal-state positional difference Δ_{t+1} to exceed a threshold δ (which is 0.05m in this study).

For the second human error correction experiment, the planning strategy is manually designed as how to sequentially correct each object according to the last section while avoiding collisions. Afterwards, the demonstrations are segmented according to objects' position with change point detection in slope [132]. The observations $\mathcal{O}$ include three raw objects' 3-dimensional positions x, 3-Dimensinoal end-effector positions and 4 task labels, while yielding a total of 16 observations. As a result, the observations can be labelled into different classes accordingly. For each cuboid manipulation, they are labelled as picking or placing stage including: 0. Picking Blue, 1. Placing Blue, 2.

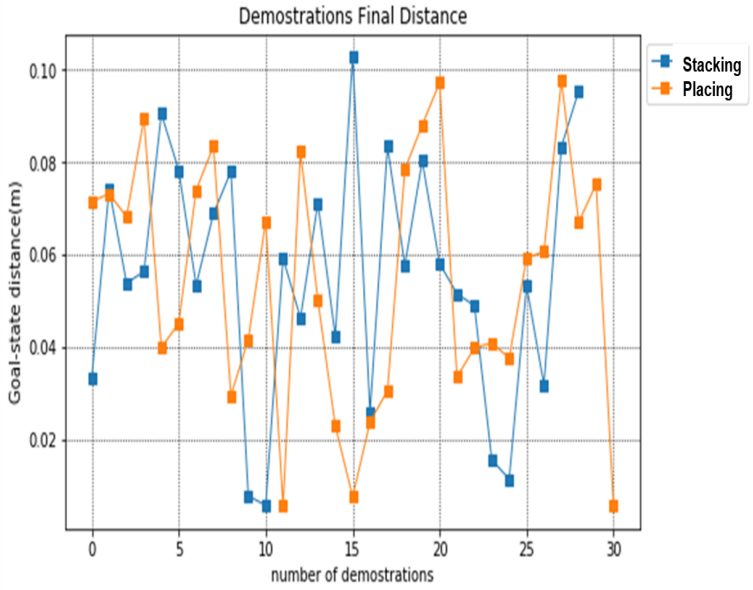


Figure 3.8: Absolute Goal-state positional difference of successful demonstrations for stacking and picking experiments collected using keyboard teleportation. For the stacking experiment, there are 28 demonstrations with a maximum difference of 0.12m. The placing experiment contains 30 demonstrations with a maximum error of 0.1m.

Picking Red and 3. Placing Red. For every stage, the time series data is re-sampled in order to keep their time length equal to 5. Overall, there are 20 demonstrations for every case in these 2 experiments.

For both experiments above, the standard DDPG with HER architecture is adopted as the baseline DRL agent. As mentioned above, it can be found that it is tricky for an RL agent to achieve a goal state formed as three object positions (shown as Fig.3.5). Instead, the goal state is assigned according to task classifier $\boldsymbol{c}$. Thus, the goal of the RL agent is to pick and place a blue object when label=0 or 1 or a red object with label=2 or 3. The goal can be expressed as the correct position and yaw angle of the corresponding object $P_{goal} = [x_{goal}, y_{goal}, z_{goal}, 0]$ and observe its current condition $P_{obj} = [x_{obj}, y_{obj}, z_{obj}, \psi_{yaw}]$. The rest of the observations include joint angle, end-effector's position and orientation. The reward is designed as a sparse reward with Euclidean distance as Eq. 3.6, where δ_r is the distance threshold. The positions are measured in meters and ψ_{yaw} is measured in radians. The actions are

$[q'_{base}, q'_{shoulder}, q'_{elbow}, q'_{wrist3}]$ within $[-6, 6]$. Meanwhile, the end-effector is disabled once any collision happens. This is to prevent the agent from learning any unnecessary behaviour.

$$R_t = \begin{cases} 0 & |P_{obj} - P_{goal}| < \delta_r \\ -1 & otherwise \end{cases} \tag{3.6}$$

During testing, the performances of the Task-conditioned subgoal planner and the DRL agents are both assessed with Success Rate (SR). At the initial stage, the objects are randomly placed within a specific range. One successful trial is defined as the final reward R is 0; thus, SR stands for the percentage of the successful trial out of the total attempts.

$$success = \begin{cases} 1 & R = 0 \\ 0 & otherwise \end{cases} \tag{3.7}$$

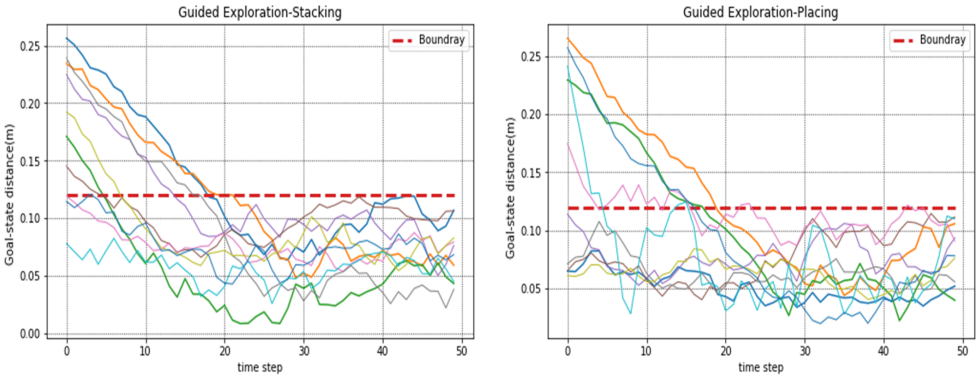
3.4 Results

3.4.1 Neural Dynamic Planner

This section focuses on evaluating the effectiveness of modelling only the expert preference, specifically the distance between the current state and the sub-goal, to generate efficient actions in the robot's joints.

For the stacking and placing experiment, Fig.3.9 illustrates the Neural Dynamic Planner's performance in guiding the early stage of DRL exploration via only direction output. As the figure shows, the DRL agent produces random actions at the exploration phase, the Neural Dynamic Planner can correct the actions, and make the goal-state positional difference coverage smaller than 0.15m.

In the human error correction experiment, the actions generated by the Neural Dynamic Planner are depicted in Fig.3.10. When the model's inputs include the complete observation, which consists of the current end-effector position and the sub-goal,



(a) Stacking. (b) Placing.

Figure 3.9: Goal-state positional difference with Guided Exploration.

denoted as $\pi(r_t, s_t)$, the model becomes a simple actor-network as those in actor-critic-based reinforcement learning methods like Deep Deterministic Policy Gradient (DDPG) [17]. These models undergo training with 5-fold cross-validation and a learning rate of 1×10^{-3}. In the experiment, the robot is permitted a maximum of 30 steps to reach the predicted sub-goal. The absolute mean distance between the end-effector and the sub-goal is recorded at each timestep.

As depicted in Fig.3.10, when full observations $\pi(r_t, s_t)$ are used, they fail to produce optimal actions, as indicated by the red curve which shows that the predicted actions remain constant. The inefficiency in state-action pairs can lead it to become trapped in local optima. Additionally, the similarity between observations may pose challenges during the training process. In contrast, the proposed Neural Dynamic Planner, represented by the blue line in Fig. 3.10, demonstrates its efficiency and adaptability by converging to a distance of less than 0.02m in approximately 15 steps.

3.4.2 Task-Conditioned Subgoal Planner

This section aims to examine the ability of the proposed subgoal planner in human error corrections with full observations without DRL. For each cuboid manipulation, the observations are labelled into different picking or placing stages including: 0. Picking

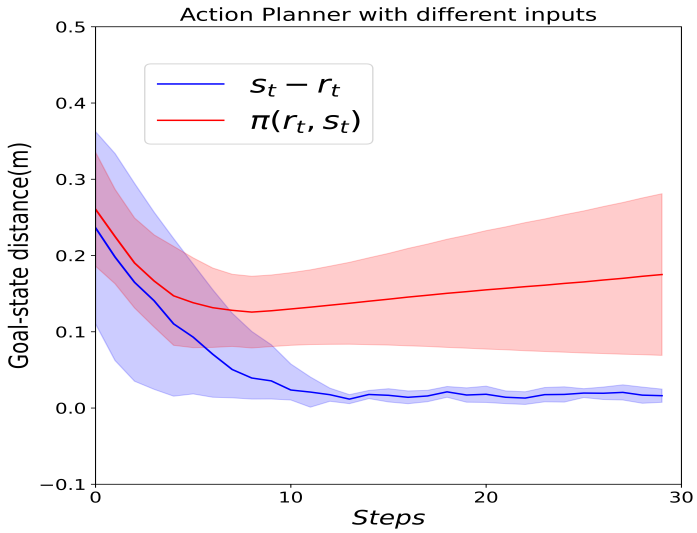
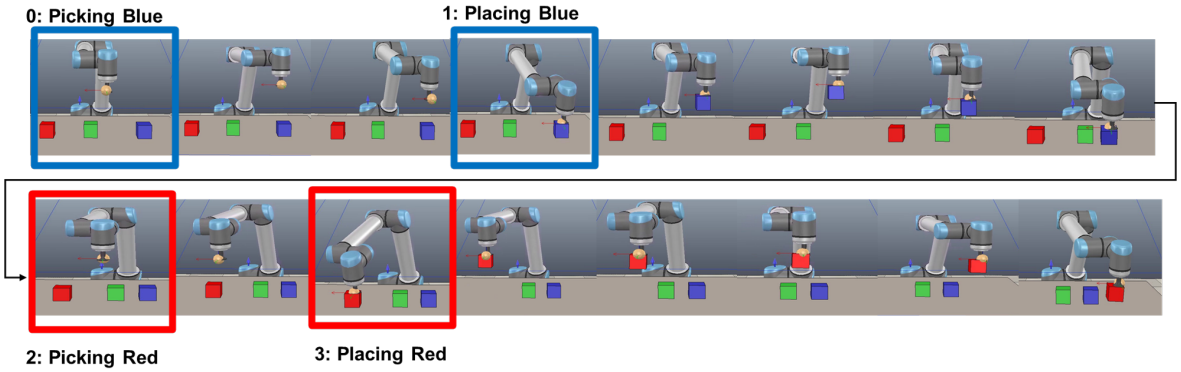


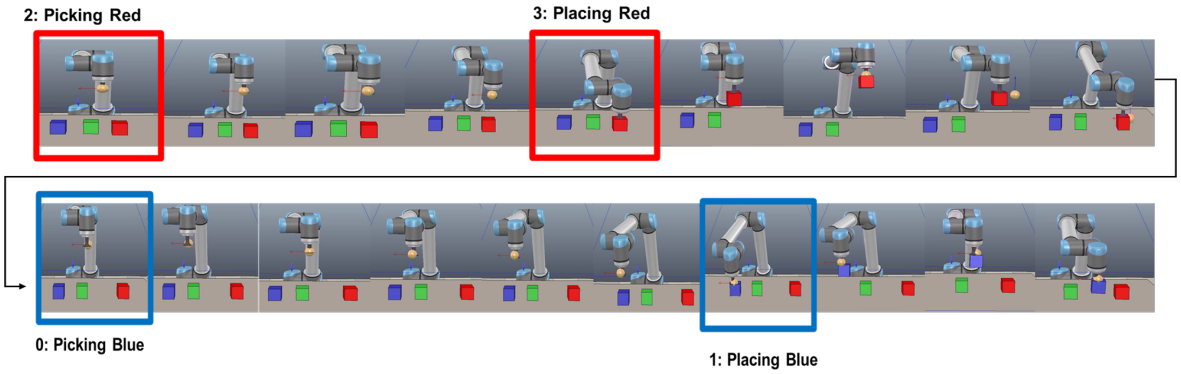
Figure 3.10: The comparison of different inputs is illustrated in the figure, which displays the absolute mean distance between the current state and the subgoal along with the standard deviation over steps. This analysis is based on 20 trials, each with random initial positions and sub-goals.

Blue, 1. Placing Blue, 2. Picking Red and 3. Placing Red. For every stage, there will be five subgoals.

Fig.3.11 describes the example frames on handling two objects correction with four task labels. It is able to adaptively arrange subgoals followed by informable actions. It also indicates that the planner is suffice to avoid any collision among objects. For the numerical results from Table 3.1, they are assisted with same reward function as Eq.3.6 with greater thresholds. It can be found the planner is not biased as the success rates of different sub-tasks in the same experiment are almost identical. Meanwhile, the success rate does not decrease a lot along with increases in the complexity of experiments, which proves the robustness of the model. However, it can be noticed that the time steps increase in some of the two object correction tasks as the predicted subgoal oscillates during the picking stage as shown in Fig.3.11b. Lastly, the main failure case is caused when the subgoal or action planned is not sufficient to pick up the object. The possible reason could be that it is not very efficient in handling large



(a) Case 1: Robot corrects the blue object as it occupies the red one's position and thus the red one sequentially.



(b) Case 2: Robot corrects the red object as it occupies the blue one's position and thus the blue one sequentially.

Figure 3.11: The different planning scenarios of two objects correction experiment as shown in Fig.3.7. The proposed method first classifies the task labels according to pre-defined goals (Fig.3.5). The robot thus corrects the corresponding objects while avoiding any collisions with the proposed two-level motion planner. The squares stand for the classified stages of picking or placing. The orange sphere in the scenes stands for predicted subgoals.

One object correction					
Task	Object	Avg. Success rates		Avg. Steps taken	
		δ_r=0.08	δ_r=0.06	δ_r=0.08	δ_r=0.06
Case 1	Blue	0.66(±0.01)	0.51(±0.05)	40.35(±4.16)	43.35(±3.73)
Case 2	Red	0.63(±0.04)	0.53(±0.02)	41.57(±3.31)	42.47(±3.2)

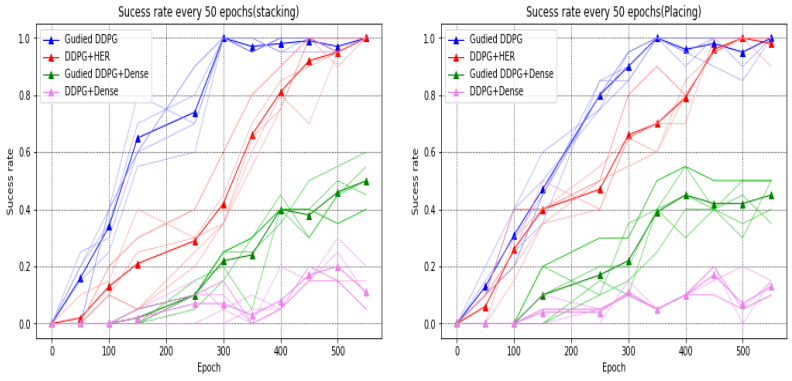
Two objects correction					
Task	Object	Avg. Success rates		Avg. Steps taken	
		δ_r=0.08	δ_r=0.06	δ_r=0.08	δ_r=0.06
Case 1	Blue	0.64 (±0.01)	0.54(±0.03)	41.35(±2.16)	48.35(±4.61)
	Red	0.61(±0.02)	0.51(±0.04)	43.11(±3.31)	48.11(±5.17)
Case 2	Red	0.59(±0.06)	0.53(±0.02)	39.53(±6.16)	41.1(±2.14)
	Blue	0.62(±0.01)	0.53(±0.04)	51.37(±5.1)	54.52(±5.1)

Table 3.1: The success rates of two-level Planner under different thresholds in two experiments.

continuous observations.

3.4.3 The importance of demonstrations for DRL

For the first stacking and placing experiments as shown in Fig.3.12. The Neural Network Planner is capable of guiding the learning of the DRL even with dense rewards by narrowing down the search space with only directions.



(a) Stacking. (b) Placing

Figure 3.12: Success rate in two tasks. Bold lines indicate the mean success rate while a lighter line represents each training curve.

Furthermore, the proposed hybrid hierarchical guidance framework is assessed for more complex human error correction tasks. In RL training, the threshold is set as $\delta_r = 0.01$ and the max step is 40. Fig.3.13 shows the training curves of total success rates of picking and placing either red or blue cuboids with the hybrid approach ($\lambda = 0.7$), fully guided RL (λ=1) and pure exploration (λ=0) for these two error correction experiments. Each subtask was trained in sequence. The overall success rates of the proposed approach can achieve 0.95 and 0.91 within 550 epochs without any collision. It can be found that the Hybrid Hierarchical Guidance is able to not only provide informative actions but also maintain diversity in the experience buffer. For full guidance, the agent is biased toward one specific task. As the agent only receives a nearly optimal strategy, the prediction accuracy for less common states or behaviours may become worse due to overfitting. On the contrary, pure exploration is sample inefficiency.

Figure 3.13: Training curves including hybrid guidance,fully guidance, and pure exploration for two experiments

3.4.4 Summary

This section discusses the proposed architecture and revisits the research questions raised in section 3.1 based on the results from the experiments conducted.

As the continuous state-action space can be quite large for robot trajectory learning, only modelling the dynamic of expert preference as the goal-state positional difference

can be efficient for dealing with object position variations. Moreover, the proposed Neural Dynamic Planner can also deal with imperfect demonstrations. As shown in Fig.3.9, it is also capable of guiding random actions with only direction modelling. This addresses $Q1$.

Moreover, the subgoal planner is able to produce collision-free subgoals through raw observations. However, high-dimensional observations affect the performance of subgoal generation.

To address **Q3**, the demonstration can indeed improve the sample efficiency of DRL. However, combining the results in Fig.3.12 and 3.13, it can be found that, maintaining the experience diversity of the memory buffer is also necessary.

However, the weakness of this work is that both the proposed planner and DRL agent suffer from high-dimensional observations and thus lead to inefficient performance. Chapter 4 aims to tackle this issue with a more complex setting further.

Chapter 4

Adaptive Task and Motion Planning in varying scenarios

4.1 Introduction

As discussed in the previous section, to tackle the robot learning problem, especially in multi-object scenarios, both the LfD-based motion planner and DRL suffer from high-dimensional observations, while the DRL still needs a long time of training even with a proper design of the guidance from the proposed motion planner. Thus, this section mainly focused on designing a pure LfD framework that can handle multi-object manipulation problems that consider variant goals and task structures. Meanwhile, this problem is termed as massive Package-to-Order (mPTO) in a manufacturing setting.

An example of mPTO (make-to-order production) is seen in packaging companies that assemble hamper baskets for their employees during festive periods. This study focuses on mPTO scenarios where the types of products are common across different customers, but the final packaged product varies based on customer specifications, as shown in Fig.4.1. This chapter addresses a couple of variations in the mPTO scenario.

- Position variations: This refers to the changing positions of objects that the robot needs to manipulate.

- Goal configurations variations: This variation is related to different customer orders, like the ones illustrated in Fig.4.1. Due to diverse customer requirements, the packaging configurations and appearances might change, even when using the same products. This means a product could be placed in various goal positions depending on the order.

- Task structure variations: This variation occurs in scenarios where humans collaborate flexibly with robots. Different humans might choose different sequences to complete a task, so the robot should be adaptable. For example, if a human starts packaging some products in a random order, the robot should be able to observe and understand what has been done so far and then continue manipulating the remaining objects in response to the human's actions. This type of collaboration is known as sequential Human-Robot Collaboration (HRC).

To address these variations in mPTO scenarios, traditional methods typically involve extensive manual planning and design. This process includes accounting for all potential variations that might occur. For instance, when using manipulation trees for planning, it's necessary to create a comprehensive list of all possible product combinations, as discussed in [57]. Similarly, logic operations in planning need to be conditioned on every conceivable action. These approaches require significant foresight and a detailed understanding of the task environment, making them complex and time-consuming.

This work utilizes the Task and Motion Planning (TAMP) framework, which offers the ability to seamlessly integrate discrete high-level decision-making with continuous motion generation.

Compared to traditional approaches, the primary objective of high-level task planning in this context is to enable a robot to learn task structures without the need for hand-coded task descriptions or predefined rules. Instead, the robot acquires knowledge of the underlying task policy solely from passive observations derived from expert demonstrations.

Figure 4.1: A real-world example of a package-to-order scenario can be seen with Amazon sellers, who frequently need to package a diverse range of products. These products must be arranged in different goal configurations to meet the specific demands of various customers. Each order might require a unique combination or arrangement of items, reflecting the individual preferences or needs of the customer placing the order. This situation exemplifies the complexity and variability inherent in package-to-order operations, where customization and adaptability are key to meeting customer expectations.

Furthermore, there are three research questions that should be addressed for the TAMP framework:

- **Q1** : In the context of mPTO scenarios, the challenge lies in how a high-level decision-making module can generate accurate sequential plans while handling redundant objects within a task structure. It's crucial to note that these redundant objects serve a purpose as they may be required for subsequent customer orders with different goal configurations. This approach has particular significance in manufacturing systems, as it can potentially reduce or eliminate downtime during changeovers.

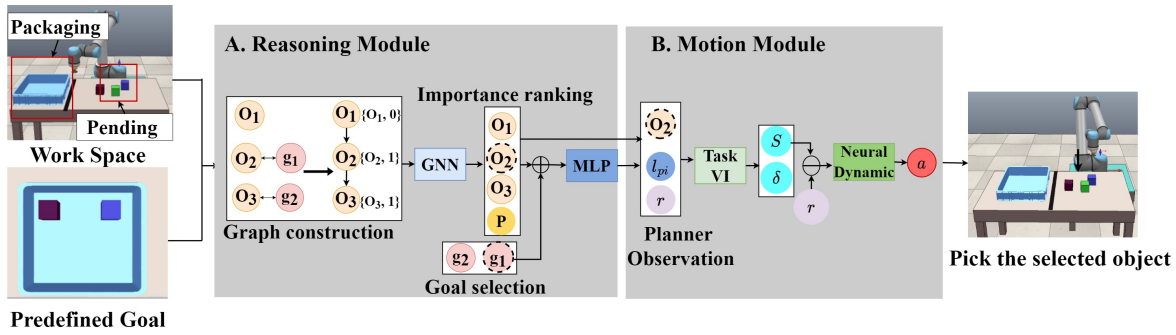
- **Q2**: How to translate the change in sequential plan to the lower level motion planner for subsequent rapid motion generation?

- **Q3**: Furthermore, in the event that a human worker becomes involved and chooses to diverge from the originally demonstrated task structure, what methods can be implemented to develop a Task and Motion Planning (TAMP) framework that can promptly detect this deviation and adapt the sequential plan accordingly?

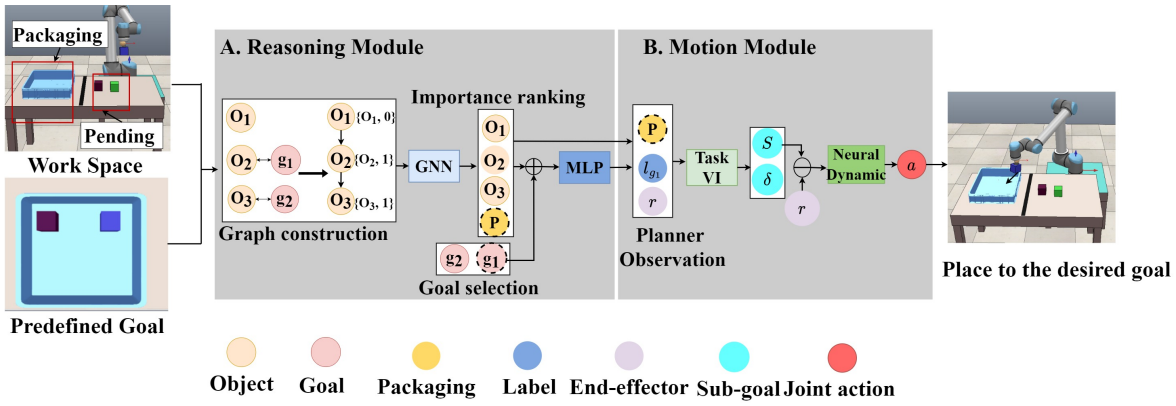
To address these issues, this chapter proposed an end-to-end architecture as follows:

- A high-level reasoning module based on Graph Neural Networks (GNN) is introduced. It aims to reason the importance of observations during different task stages. In comparison to existing GNN-based methods, this approach offers more detailed guidance for planning low-level trajectories. Differing from prior work in [74], this method is capable of generating distinct task outlines during both the picking and placing stages. Furthermore, in [74] and [77], extensive information about object types, object fulfilment (i.e., whether objects have reached their intended positions), and goal types are required. Consequently, their high-level GNN models still demand additional effort to define conditions like if-else statements. The aim of this work is to alleviate designers from this complexity by utilizing only positional data and essential object features in the reasoning module.

- For low-level motions, the two-level motion planner proposed in the previous Chapter is adopted to integrate and imitate various trajectories from experts. Unlike using full observations, the inputs have been effectively reduced through the use of the high-level reasoning module. Moreover, this Chapter provides a more extensive study of the proposed motion planner. As shown in the experimental session, the motion planner can produce new collision-free trajectories when novel goal positions have been perceived.



(a) Picking stage



(b) Placing stage

Figure 4.2: A graphical representation of the proposed framework is presented during both the picking and placing stages. During the picking stage, the reasoning module consistently directs its attention to the selected objects within the pending area. In contrast, during the placing stage, it consistently focuses on the observations of the packaging box.

4.2 Methodology

Its objective is to offer an end-to-end solution learned from expert demonstrations for mPTO scenarios, following a Procedural Attachment approach. It operates hierarchically, determining the allocation of important features across various levels of planning.

Suppose the obtained demonstrated observations can be expressed as a tuple $\Pi = \{g, o, p, r, \mu, a, I\}$ from environment scenes. There will be m products with position information $o = \{o_i\}_{i=1}^{i=m}$ from the pending area. According to customer demand, there will be n selected products with goal position information $g = \{g_i\}_{i=1}^{i=n}$ at the packaging box p $(n \leq m)$.

The robot is tasked with sequentially transporting selected products to the packaging area, with consideration for two distinct cases encompassed within the same framework:

1. **CASE 1**: the robot operates independently by following the same task structure learnt from demonstrations. Here, the robot engages in a sequential process of picking and placing products, one after the other, until the predefined goal configuration is attained.

2. **CASE 2**: in the second case, the robot encounters different task structures, particularly when HRC comes in. In such instances, human actions can exhibit variability, such as employing diverse arm movement trajectories for picking and placing objects. Subsequently, this proposed approach primarily focuses on monitoring the final stages of human actions, specifically the objects' final positions.

Figure 4.2 serves as an illustration of the proposed framework, which draws inspiration from the natural human approach to object manipulation. It takes into account the common sequence in which humans typically decide to interact with objects. The observed human demonstrations are thus divided into two distinct task stages for each selected object and its subsequent manipulation:

1. At the **picking** stage, the robot's primary task is to identify and prioritize the most important object within the set of objects o and proceed to pick it up.

2. Subsequent to the object being successfully picked up, the robot's next objective is to transport the object to a designated goal pose. It is termed as the **placing** stage, which assigns priority to the packaging box position denoted as p as the most pivotal feature. The specific goal position is deduced based on the task label l, enabling the robot to complete the placement task effectively.

4.2.1 Reasoning Module with Graph Neural Network

In prior studies, as seen in [74] and [77], the graph representations included both object and goal nodes, with additional attributes defining object types and their fulfilment status. While the goal positions for objects could exhibit variations, especially in scenarios involving diverse geometric shapes for final goal configurations, such an approach often led to a rigid adherence to predefined task structures. In these previous works, the incorporation of additional if-else statements became necessary at each step to determine whether the goal had been fulfilled or not. In contrast, this study adopts a different approach. It seeks to enable the agent to initially infer crucial observations and stages solely through object position information, supplemented by essential features. Furthermore, the objective is to enhance the agent's adaptability in scenarios where the task structure undergoes variations.

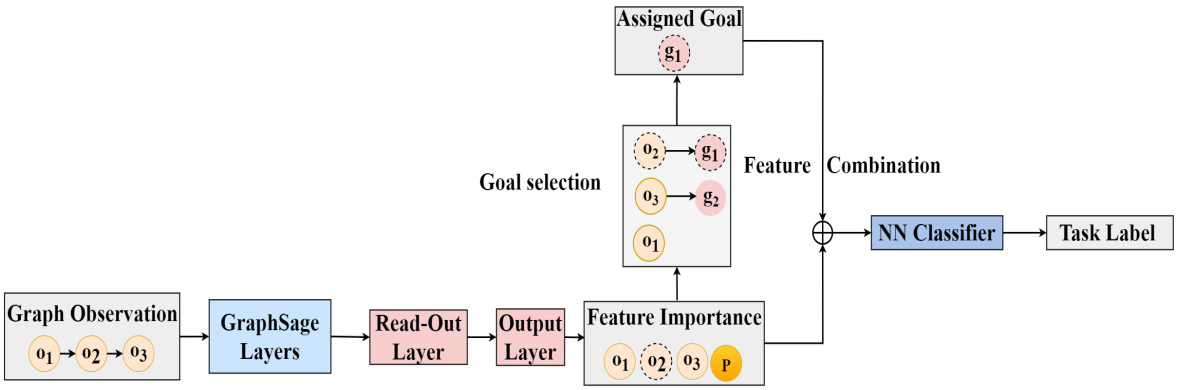


Figure 4.3: The Neural Network Architecture for the reasoning module follows this structure: A graph representation of the object states is initially constructed. Three GraphSAGE layers are applied, along with the ReLu activation function, to extract the most pertinent observations. This process results in a probability distribution that aids in selecting the corresponding goal from a set of potential goals. Subsequently, the task stage is classified by combining these two features. This classification is achieved through a 3-layer neural network, with ReLu serving as the activation function.

To achieve these goals, the main idea is to reduce the high-dimensional observations through importance ranking. It first assigns the necessary goal positions to each selected object. This will be selected by the importance score at the beginning of the

picking stage. Each single object manipulation task is treated as a graph classification problem. The output of the graph neural network will be a $m+1$ dimensional probabilistic distribution $P^o_{pred} = \{p^o_{pl}, p^o_1, ..., p^o_m\}$ where $\{p^o_1, ..p^o_m\}$ depicting the object importance within o at the picking stage and an extra p^o_{pl} suggests the importance of packaging area p at placing stage.

The distribution P^o_{pred} is combined with the selected goal position. This will then be further fed to a fully connected NN classifier for classifying $n+1$ dimensional probability distribution $P^g_{pred} = \{p^g_{pi}, p^g_1, ..., p^g_n\}$, where $\{p^g_1, ..p^g_n\}$ describes the goal labels at placing stage and an extra p^g_{pi} represents the picking stage. Note that each p^g_n in $\{p^g_1, ..p^g_n\}$ represents a specific goal position, which is demonstrated by the predefined goal configuration. It is referred to as position-specified labels. Such a design is to better infer the low-level motion module about the task stages.

Thereafter, the important features selected by P^o_{pred} and the one-hot encoded label $l \in \{l_{pi}, l_{g_1}...l_{g_n}\}$ converted from P^g_{pred} are combined together as the inputs for the motion module (See Fig.4.2) .

Fig.4.3 describes the neural network architecture for the reasoning module. GNN are introduced to operate the graph classification. The states are encoded as graphs. Let there be n out of m ($n \leq m$) objects that need to be manipulated. There will be m nodes $V = \{v_m\}^m_{m=1}$ and each node contains 4-dimensional features $\phi(v_m)$, including the 3-dimensional objects positions o and an extra binary property $I = 0$ *or* 1 describing whether such an object has been selected or not according to the predefined $g = \{g_n\}^n_{n=1}$. The directed linking edges can be expressed as $E = \{e_{i,j}\}$ for $i = 1, ..m-1$ and $j = 2, ...m$, where each node is only connected with its neighbour nodes.

This study adopts GraphSAGE(Sage) [133] layer. It holds the advantage of being generalizable to unseen nodes by sampling and aggregating the target node's neighbour nodes instead of weighting the whole neighbour nodes like Graph Convolution Network(GCN) [134].

Assume the initial node embedding is $h^0_i = \phi(v_m)$ and there will be K message passing iterations or K layers. It can thus aggregate its neighbour nodes h^{k-1}_j from the

previous layer (i.e $K-1$) and to form a single vector representation as Eq.4.1. In this study, f_{agg} is the aggregator that aggregates the neighbours' features with an averaging function $\frac{1}{N}\sum_{j\in\mathcal{N}(i)} h_j^{k-1}$. This aggregated representation $h_{\mathcal{N}(i)}^k$ will be concatenated with the target node's embedding from the previous layer h_i^{k-1} and further multiplied by a weight matrix W_k. Thus, the node embedding of K_{th} layer can be represented as Eq.4.2, where σ is the ReLu activation function. $f_{\theta_{gnn_1}}$ and $f_{\theta_{gnn_2}}$ are the trainable functions with parameters θ_{gnn_1} and θ_{gnn_2} for each layer. In order to prevent gradient explosion, the obtained node embedding are normalized as $h_i^k \leftarrow \frac{h_i^k}{||h_i^k||_2}$

$$h_{\mathcal{N}(i)}^k = f_{agg}(h_j^{k-1}, j \in \mathcal{N}(i)) \tag{4.1}$$

$$h_i^k = \sigma(W_k \cdot (f_{\theta_{gnn_{k-1}}}(h_i^{k-1}) + f_{\theta_{gnn_k}}(h_{\mathcal{N}(i)}^k)) \tag{4.2}$$

Afterwards, for graph classification, there will be an additional readout layer that aggregates the node embeddings into a graph embedding as Eq. 4.3. A final output layer accepts the graph embedding and produces m+1 final categorical distribution P_{pred}^o.

$$G_k = \frac{1}{\mathcal{N}(v)} \sum_{i\in\mathcal{N}(v)} h_i^k \tag{4.3}$$

A classifier is further built with three layers. It takes m+1 dimensional distribution P_{pred}^o and a 3D selected goal position g as inputs, which yields total m+4 dimensional features with the final outputs P_{pred}^g.

For training this module, it is considered as a supervised learning model with the ground truth distribution P_{goal}^o and P_{goal}^g. For both GNN and NN classification, the cross-entropy loss is used as Eq.4.4 and 4.5. These two cost functions are jointly

optimized as a linear combined cost function in Eq.4.6.

$$loss_1 = -\sum_{m=1}^{m} [p^o_{goal}]_m log(p^0_{pred})_m \tag{4.4}$$

$$loss_2 = -\sum_{n=1}^{n} [p^g_{goal}]_n log(p^g_{goal})_n \tag{4.5}$$

$$\mathcal{L}_{re} = (loss_1) + (loss_2) \tag{4.6}$$

4.2.2 Motion module

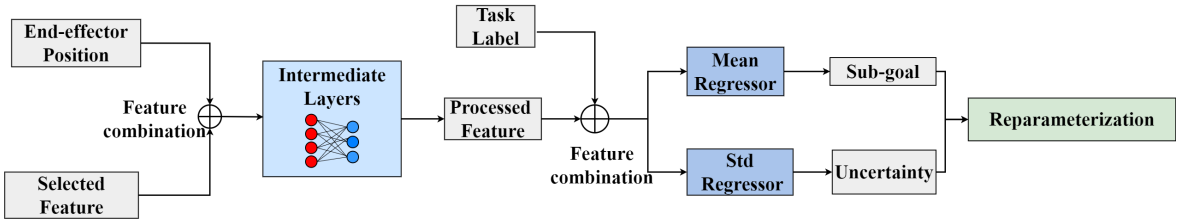


Figure 4.4: Neural Network for the sub-goal planner with ranked inputs.

This section focuses on building a motion module for a robot to generate actions based on the information provided by the reasoning module. In mPTO scenarios, collision-free motions need to be considered in order to avoid any damage to the products. Thus, a robot should avoid any collisions with itself and with objects at the picking stage. It should also avoid collisions between various objects already packed during the placing stage.

The methodology of the motion module is the same as the one described in the previous section, while the inputs of the subgoal planner are different as shown in Fig.4.4.

Since the reasoning module can effectively reduce the observation dimensions, the subgoal planner in this study is used to produce adaptive plans in dealing with more variant picking and placing positions. In detail, the sub-goal plan s is based on the current end-effector pose r and different features obtained from the reasoning module

at different stages. In the picking stage, the reasoning module will always provide the selected object position $o_{selected}$ and label indicating the task stage l_{pi}, and thus lead to total observations $\mathcal{O} = \{o_{selected}, l_{pi}, r\}$. During the placing stage, the reasoning module will focus on the packaging box position p with a label describing different goal positions l_{g_i}, and lead to $\mathcal{O} = \{p, l_{g_i}, r\}$. By accessing the task information (e.g. label), it is able to provide 3-dimensional collision-free sub-goals for different target poses even when similar observations are perceived.

4.2.3 Task and Motion Planning framework

In the testing phase, the high-level planning process commences by having the reasoning module identify both the task label and the most critical observation. Subsequently, in the motion module, the task-conditioned sub-goal planner utilizes the information supplied by the reasoning module to propose conditioned mean sub-goals. Simultaneously, the neural dynamic planner is responsible for executing the low-level actions aimed at achieving the specified sub-goals, as outlined in Algorithm 2.

Algorithm 2 Proposed TAMP architecture for adaptive packaging problem

Initial observations $\Pi = \{g, o, p, r, \mu, a\}$

Trained reasoning module $\mathcal{R}$, and motion module with **task-conditioned sub-goal planner $\mathcal{S}$,neural dynamic planner $\mathcal{D}$**

In picking stage

Construct graph G based on objects o and their assigned goals g

Produce the important object $o_{selected}$ and select its goal $g_{selected}$ from GNN, and thus obtain the task label for picking l_{pi} as $\{o_{selected}, l_{pi}\} = \mathcal{R}(G, g_{selected})$

for *every sub-goal planning step in picking* **do**

  Produce sub-goal $s_t = \mathcal{S}(o_{selected}, r, l_{pi})$

  for *every action planning step* **do**

    Produce joint actions $a_t = D(\Delta_t)$, where Δ_t is the distance between end-effector position and sub-goal $\Delta_t = s_t - r_t$.

In placing stage

Construct graph G based on objects o and their assigned goals g

Produce the packaging box position p from GNN and the placing goal label l_{g_i} as $\{p, l_{g_i}\} = \mathcal{R}(G, g_{selected})$.

for *every sub-goal planning step in placing stage* **do**

  Produce sub-goal $s_t = \mathcal{S}(p, r, l_{g_i})$

  for *every action planning step* **do**

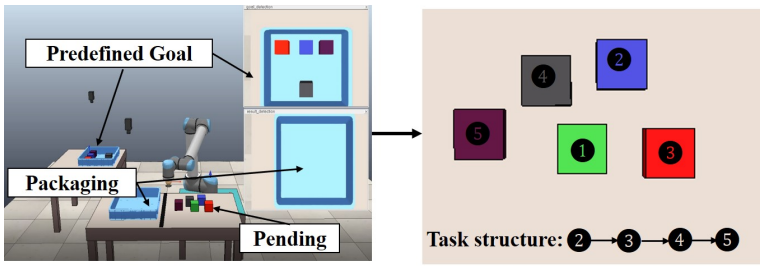
    Produce joint actions $a_t = D(\Delta_t)$, where Δ_t is the distance between end-effector position and sub-goal $\Delta_t = s_t - r_t$.

4.3 Experimental Setup

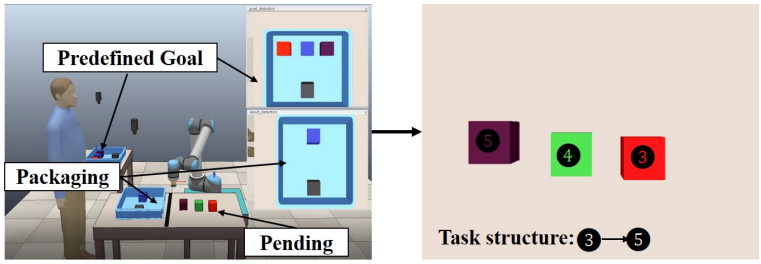
In this study, the use cases have been devised to emulate scenarios resembling mPTO challenges. However, due to the constraints related to the absence of extensive customer orders, these use cases have been narrowed down to an adaptive packaging problem. This adaptation still retains the essence of the changes typically encountered between batches of large-scale orders.

The study encompasses a total of n out of m objects ($n \leq m$) for the adaptive packaging experiments. An illustrative example is provided in Figure 4.5, representing a scenario with 4 out of 5 selected objects. Each experiment corresponds to a specific customer order, specifying the requirement to package n objects from a set of m objects.

These experiments introduce a variety of goal configurations, as depicted in Figure 4.6, where each object possesses the potential to be selected. The number of possible combinations for such configurations is calculated using ${}_mC_n = \frac{m!}{(m-n)!n!}$. Additionally, while the positions within the goal configuration remain fixed, the same objects may assume different positions within the goal configuration. Therefore, for each combination, there are $P_n = n!$ possible permutations. Consequently, every experiment entails a total of ${}_mC_n \times P_n$ distinct goal configurations.

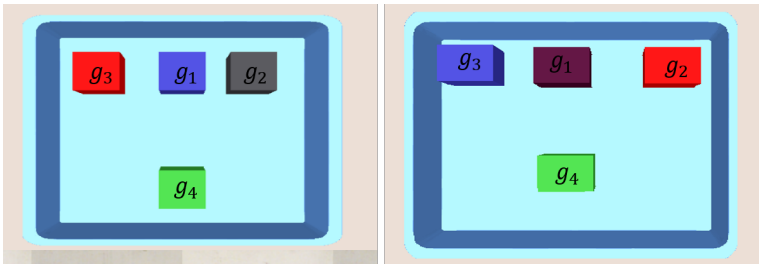


(a) **CASE 1**: The simulation experiment in 4 out of 5 experiment with the task structure followed by the demonstrator

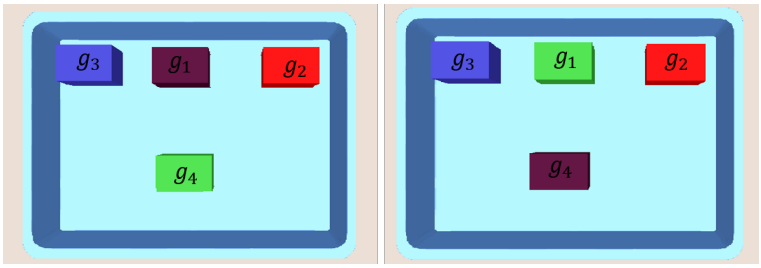


(b) **CASE 2**:human manipulate objects randomly, and the robot should generalize to different task structures to carry out the rest of the task.

Figure 4.5: This figure illustrates two distinct scenarios: **CASE 1**, where the robot adheres to a fixed task structure, and **CASE 2**, where the robot collaborates with a human, engaging in a variable task structure. In both scenarios, the robot's objective is to pick and place the designated objects into the blue packaging box from the pending area. Please note that the human model featured in the figures serves purely illustrative purposes.



(a) Different combinations in goal configurations when selecting 4 objects among 5 objects



(b) Different permutations under the same combination

Figure 4.6: These figures show various goal configurations, which include possible combinations and permutations, along with position-specified labels. Figure 4.6a shows the packaging order needs different objects (combinations). Figure.4.6b shows the packaging order need the same objects while with different appearance (permutations).

The objective of this study is to address two distinct scenarios within each experiment, all within the framework of the proposed architecture. These two cases are as follows:

- **CASE 1**: In this scenario, the robot is tasked with managing the varied goal configurations independently, as described in Figure 4.5a. The robot's approach aligns with the learned task structure obtained from demonstrations. For instance, if the selected objects are [2,3,4,5], the robot consistently manipulates the selected object with the lowest numerical identifier until it achieves the final goal.

- **CASE 2**: In contrast, Case 2 serves as a demonstration of the generalization capabilities inherent in the proposed approach, particularly when confronted with unforeseen task structures arising from human actions and diverse goal configurations. For example, as illustrated in Figure 4.5b, the human intervenes by picking

the first and third selected objects, leaving the robot with the task of sequentially manipulating the second and fourth selected objects. This case assesses the adaptability of the proposed framework to accommodate varying human performance and task structures.

To train the architecture, the high-level reasoning module and the low-level motion module are trained separately using different expert demonstrations.

In the high-level reasoning module, the desired goal configuration is initially demonstrated, utilizing position-specified labels, as depicted in Figure 4.6. Consequently, experts manipulate the objects in accordance with the established task structure as presented in Figure 4.5a. During this process, only two graph-based observations are gathered for each instance of single-object manipulation, occurring at the commencement of both the picking and placing stages, as visualized in Figure 4.9. To facilitate the learning process, ground truth distributions denoted as $P^o goal$ and $P^g goal$ are provided in the form of one-hot vector labels. This ensures that only the probability selected by the expert is assigned a value of 1, while all other probabilities are set to 0. Subsequently, the agent learns the underlying structures from graph-encoded observations, all without the need for manually defined task descriptions or rules.

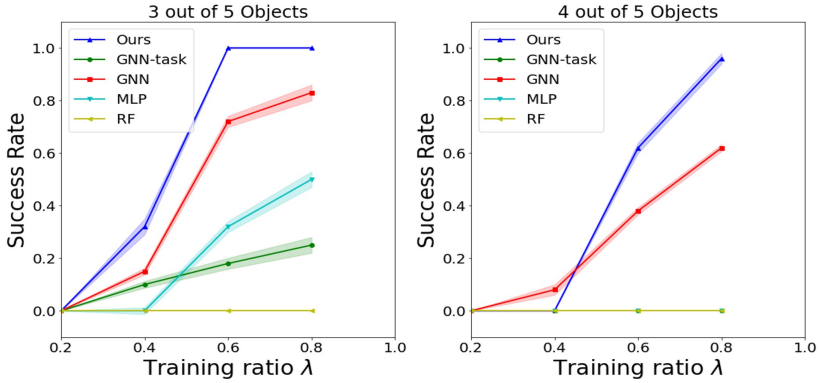
In the low-level motion module demonstrations, the robot is operated manually by an expert to perform the tasks of picking and placing objects. Three distinct sub-goals are designed to govern various stages of the process. For the picking stage, the robot executes a sequence of actions as follows: it first positions itself above the target object, followed by a controlled approach toward the surface of the object, and ultimately, it performs the action of grabbing and lifting the object. In the placing stage, the robot adheres to a similar sequence of actions: it initially approaches a specific point relative to the goal position, then moves above the goal, and finally carries out the action of placing the object in its desired location.

Each of these sub-goals encompasses a set of 20 distinct actions. These actions are recorded and include the three most impactful joint actions of the UR10 robot,

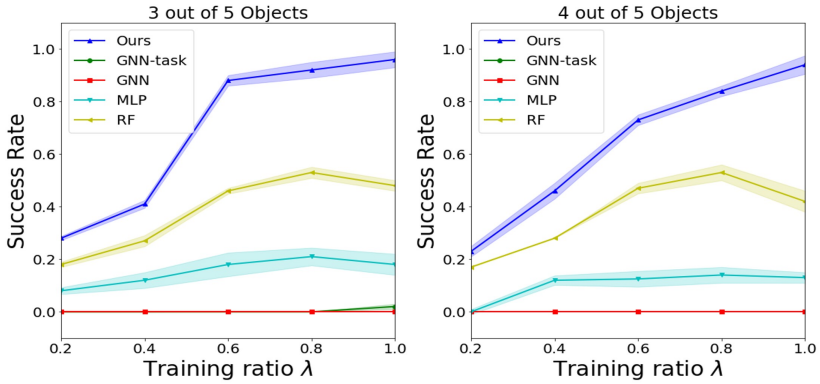
specifically denoted as $[q'_{base}, q'_{shoulder}, q'_{elbow},]$.

4.4 Results

4.4.1 Reasoning Module



(a) Generalization study on handling unseen goal configurations in **CASE 1**



(b) Generalization study on handling unseen task task structures and goal configurations in **CASE 2**

Figure 4.7: An investigation into the generalization performance across different training ratios was conducted in a simulated environment.

This section's primary objective is to demonstrate the efficacy of the GNN-based reasoning module in managing a substantial volume of goal configurations that involve redundant objects. Additionally, it seeks to assess the module's capacity for generalization when confronted with previously unseen goal configurations and task structures

across various task stages. To achieve this, the reasoning module is subjected to comparison against alternative methods within the context of the mPTO use case:

1. **GNN**[74]: The GNN-based high-level policy from the work [74] with GraphSAGE Layers is implemented. The observations are encoded as a fully connected graph. This graph provides both objects and goals nodes with spatial 3D positions and extra properties regarding object types and fulfilment in the graph. [74] performs the graph classification to directly output the two probability distribution of the target object P^o_{pred} and goal P^g_{pred}. In order to meet the needs of the experiment, there are extra binary features as I describing whether an object has been selected or not in every node. Their work can handle various positions for each object's goal by producing the same distribution P^g_{pred}. Hence, this ground truth P^g_{goal} label does not reflect the specific goal position. Instead, it only represents the order of goals. Furthermore, the same ground truth for picking and placing stages is assigned as their work only considers the initial observation at the beginning of each object manipulation.

2. **GNN-task**: A design like **GNN** cannot distinguish between different goal positions. Hence the low-level motion module will always follow the same static trajectory regardless of the goal positions. In the comparisons that follow, their approach is trained with ground truth P^o_{pred} and P^g_{goal}.

3. **MLP**: In comparing the approach with a traditional Multi-Layer Perceptron (**MLP**), the input is flat 1D observations instead of graphs. This was trained with expert demonstration data.

4. **RF**: The proposed approach is compared with a Random Forest Classifier (**RF**) which is a Neural Network-based traditional approach. This approach infers the target predictions by using an ensemble of decision trees with each tree containing a sub-sample of data features. Each tree has branches that use Boolean-type logic

to reach a decision. The **RF** reaches a decision through a majority vote from an ensemble of trees.

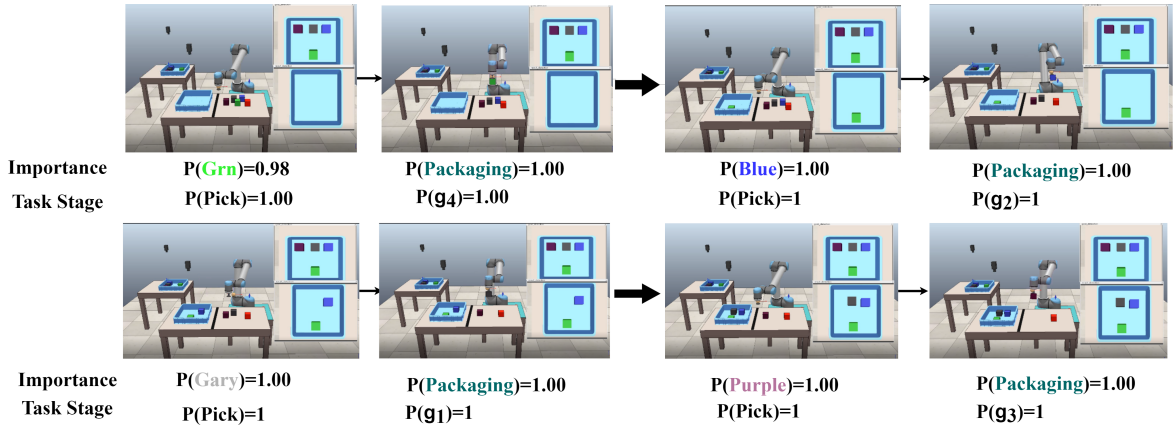
The training of these methods involves a 5-fold cross-validation setup with a learning rate as 1×10^{-3}. During the testing phase, the assessment relies on a Success Rate (SR) metric, which measures the percentage of successful trials out of the total attempts. The classification results are derived for P^o_{pred} and P^g_{pred} at both the picking and placing stages. This evaluation is conducted for every individual object manipulation, and a trial is deemed successful when all predictions for the selected n objects are accurate. To illustrate, in a single experiment involving n out of m objects, there will be a total of n x 2 predictions for both P^o_{pred} and P^g_{pred}.

n out of m	2 out of 3	3 out of 3	3 out of 5	4 out of 5
Ours	1 ± 0.000	1 ± 0.000	1 ± 0.000	1 ± 0.000
GNN	1 ± 0.000	1 ± 0.000	1 ± 0.000	1 ± 0.000
GNN-task	0.83 ± 0.015	0.72 ± 0.017	0.58 ± 0.024	0.24 ± 0.031
MLP	1 ± 0.000	1 ± 0.000	0.96 ± 0.013	0.86 ± 0.025
RF	1 ± 0.000	1 ± 0.000	1 ± 0.000	1 ± 0.000

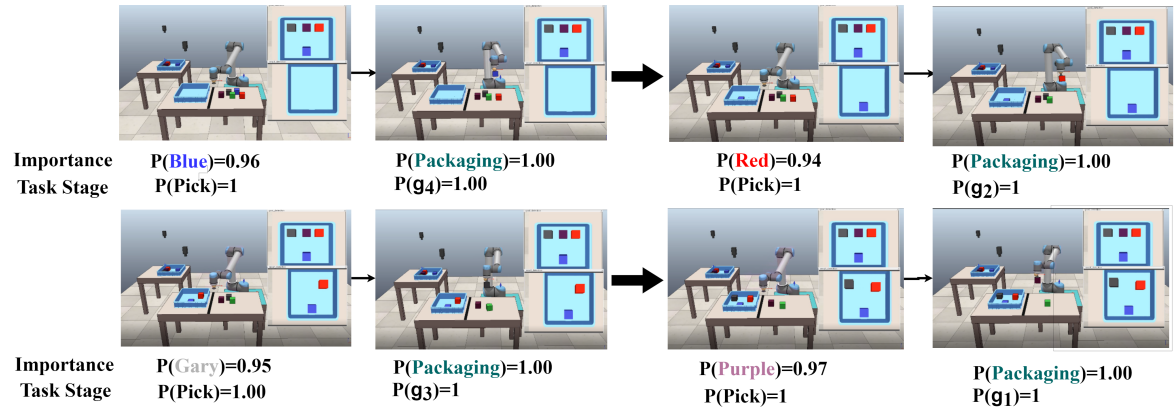
Table 4.1: Success rates of different methodologies for reasoning with n out of m objects in simulation.

Table 4.1 provides an overview of the performance in **CASE 1**. Each model undergoes training with the complete demo dataset, and their performance is evaluated when objects' positions are randomly initialized within the pending area. As demonstrated in Table 4.1, the proposed approach exhibits competitive performance, achieving a 100% Success Rate (SR) across a wide range of diverse scenarios. For instance, in the 4 out of 5 experiment, there are 120 different scenarios. The **RF** method also delivers a comparable performance. However, the **GNN-task** model performs the least effectively, likely due to its direct production of position-specific labels from the extracted GNN features. This underscores the effectiveness of employing an additional classifier for inferring distinct task stages, particularly when generating varied position-specific labels during the placing stage. The **MLP** method experiences a decrease in SR, which

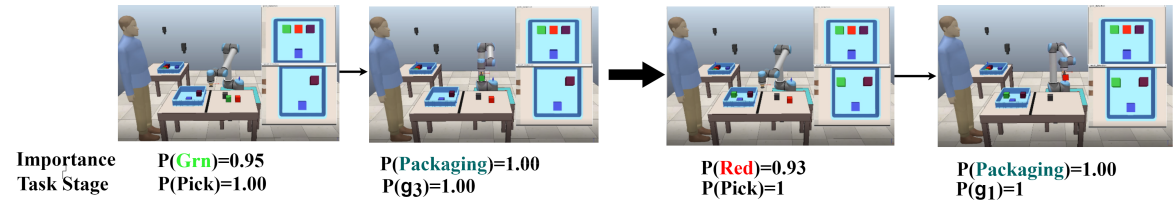
can be attributed to the challenges posed by variations in object positions.



(a) Demonstrated goal configuration



(b) Generalization on unseen goal configuration



(c) Generalization on unseen task structures

Figure 4.8: The simulation results in the context of the 4 out of 5 experiments demonstrate that the reasoning module is capable of effectively handling previously unseen tasks. Fig.4.8a depicts the performance on the demonstrated task, while it is able to handle untrained goal configuration independently in Fig.4.8b and unseen task strictures with human collaboratively in Fig.4.8c.

Furthermore, the study emphasizes the module's ability to generalize when con-

fronted with observation distributions that fall outside the training data. The modules in this study were exclusively trained using a fraction (represented as the training ratio λ) of the dataset, featuring only partial goal configurations.

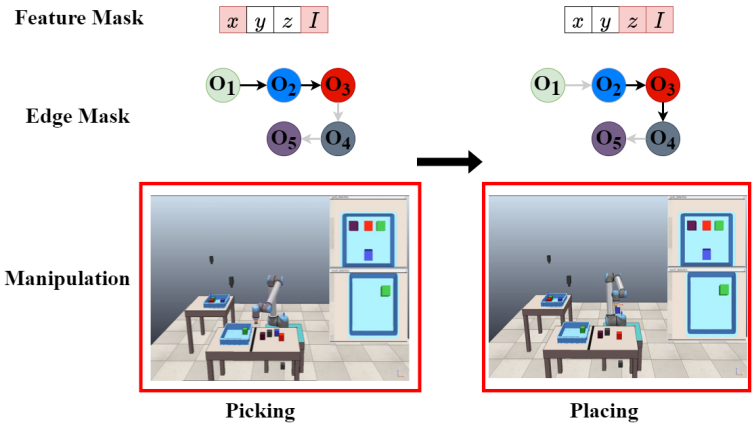
In **CASE 1**, the module selects combinations of goal configurations randomly, representing $\lambda \times_m C_n$ combinations along with their possible permutations P_n for model training. Each unique goal configuration is used only once, and the initial positions of objects are randomized. The objective of this experiment is to evaluate the model's Success Rate (SR) on unseen tasks, specifically $(1-\lambda) \times_m C_n \times P_n$ tasks during testing. This means that during testing, the module must generate task structures for previously unencountered goal configurations, including scenarios involving previously unobserved redundant objects.

Figure 4.7a shows the results for **CASE 1**. The proposed model attains a 100% SR for unseen goal configurations when $\lambda = 0.6$ in 3 out of 5 experiments. Moreover, it demonstrates the capability to handle 24 unseen goal configurations with 94 trained demonstrations in 4 out of 5 experiments. Conversely, both **GNN** and **GNN-task** fail to generalize effectively to unseen goal configurations that involve redundant objects, while **RF** encounters difficulties when generalize to previously unobserved goal configurations.

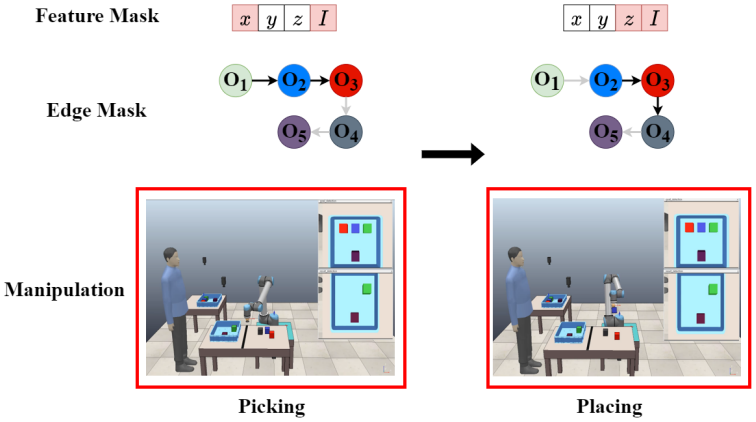
In **CASE 2**, similar to the study in **CASE 1**, the models are trained using a subset of goal configurations, specifically $\lambda \times_m C_n \times P_n$. During the testing phase, an augmented dataset is employed, simulating diverse task structures by utilizing randomly selected goal configurations (as shown in Fig. 4.5b). Each n out of m experiment comprises a total of 300 distinct testing scenarios.

CASE 2 as shown in Fig.4.7b, is challenging due to the increase in the amount of unseen scenarios. When the proposed module is trained with full goal configurations ($\lambda = 1$), it exhibits the highest average success rate (SR) of 96.3% for 3 out of 5 experiments and 93.7% for 4 out of 5 experiments in handling unseen task structures. However, **GNN** struggles to adapt to different task structures, as evidenced by its inability to produce accurate importance predictions (P^o_{pred}). Interestingly, **MLP**

outperforms **GNN** and **GNN-task**, possibly due to the risk of aggregating irrelevant neighbour node features in a fully connected graph, which can impact prediction accuracy. While RF still encounters difficulties in generalizing to diverse task structures. The simulation results, highlighting the generalization capabilities of the reasoning module for unseen goal configurations and task structures, are presented in Fig. 4.8.



(a) Case 1: Showing the robot moving towards the objects for subsequent picking and placing into a box



(b) Case 2: Showing the handling of different task structures when collaborating with a human

Figure 4.9: The interpretation of the learned reasoning module is provided for both picking and placing stages in two cases. Each sub-figure displays the following information: Feature mask (first row): Identifies the most important features. Edge mask (second row): Highlights the most critical edges using solid lines. And manipulation scenes (third row): Offers visual representations of the manipulation scenes from the simulation.

The results of the learned importance ranking are explained using GNNExplanier [135], as shown in Fig. 4.9. GNNExplanier is a tool designed to explain trained graph models by identifying the most crucial features and edges. In Fig. 4.9, the feature I proves essential in the mPTO problem, enabling the GNN to discern which subsets of objects should be prioritized. It efficiently computes object importance based on position information and neighbouring nodes during the picking stage. Additionally, the spatial feature of object height z plays a role in the placing stage. Remarkably, the model exhibits robustness in handling various task structures, with identical masks observed in both CASE 1 and CASE 2. This indicates that the graph construction empowers the trained GNN to effectively identify if some objects with $I = 1$ have been packaged (i.e., influenced by human performance) using solely position information. Consequently, it can accurately plan the remaining selected objects according to the learned task structure, as demonstrated in Fig. 4.8.

4.4.2 Motion module

4.4.2.1 Task-Conditioned Sub-goal Planner

This section intends to demonstrate the significance of supplying task information from the high-level module and evaluate the efficacy of the sub-goal planner in handling variations in object positions. The sub-goal planner, operating as a probabilistic regressor, is subjected to comparison against various regression methodologies, as detailed below.

1. **VI**: This is a variational inference regressor without any task label.

2. **GPR**[136, 137]: Gaussian Process Regression (GPR) is a non-parametric regression technique rooted in Bayesian principles. It involves training with a provided task label and employing a Radial Basis Function (RBF) kernel.

3. **MLP**: A simple neural network-based deterministic regression model with task labels (i.e. it does not measure the uncertainty of $p(s \mid \mathcal{O})$).

	2 out of 3		3 out of 3		3 out of 5		4 out of 5	
Model	R2	SR	R2	SR	R2	SR	R2	SR
Ours	0.98	0.96±0.012	0.98	0.94±0.008	0.97	0.92±0.012	0.97	0.89±0.017
VI	0.96	0	0.94	0	0.95	0	0.97	0
GPR	0.98	0.35±0.046	0.99	0.43±0.076	0.99	0.32±0.047	0.99	0.21±0.031
MLP	0.87	0.26±0.041	0.73	0.12±0.023	0.621	0.08±0.019	0.43	0.06±0.016

Table 4.2: The results of various sub-goal regression methods on the demonstration data, along with the success rates in the adaptive packaging experiments, each consisting of 100 testing trials.

The learning rates for these experiments were set at 1×10^{-3}. To assess the regression results, 3-fold cross-validation was employed, and the performance was measured using the R2 score. A higher R2 score indicates that the model can better explain the variability in the data.

The different sub-goal planners were then incorporated into the TAMP architecture and evaluated through simulation experiments. During the training phase, each object was only allowed to be manipulated to one goal position. In contrast, during testing, the generalization ability of the proposed sub-goal planner was tested when the same object needed to be used in different goal positions, such as in permutations under the same combination (as illustrated in Fig. 4.6a and Fig. 4.11b).

During testing, 100 trials were randomly selected from **CASE 1** and **CASE 2**, as discussed in the previous section. A successful manipulation of an object was defined as when the Euclidean Distance between the object's pose and the goal position was under a certain threshold $\delta_r = 0.06m$ (as in Equation 4.7). A successful trial was one in which all target objects had reached their predefined goals without any collisions during execution

$$success = \begin{cases} 1 & \mid Pose_{obj} - Pose_{goal} \mid < \delta_r \\ 0 & otherwise \end{cases} \tag{4.7}$$

The proposed task-conditioned sub-goal planner demonstrates its ability to handle up to 4 different goal positions, as indicated in Table 4.2. In contrast, the **MLP**

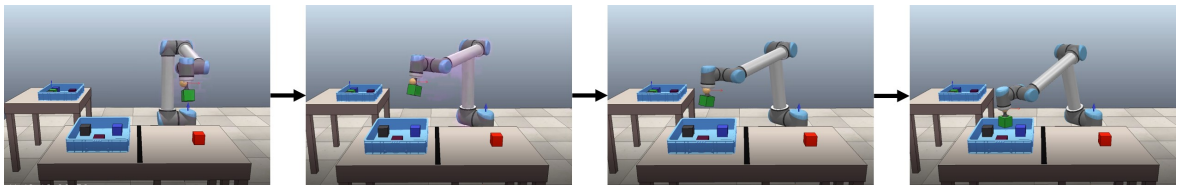
exhibits the poorest performance, limited to handling only 2 different goal positions. The **VI** model achieves a better $R2$ score within the training dataset; however, it struggles to generate stable sub-goals in the absence of task information, highlighting the significance of task labels.

On the other hand, the **GPR** model utilizes information from the entire training dataset to predict sub-goals. Despite having access to task information, it still fails to adapt to unseen goal positions that deviate from the training distribution. In contrast, the task-conditioned variational-inference-based sub-goal planner performs better in handling variations observed during testing experiments.

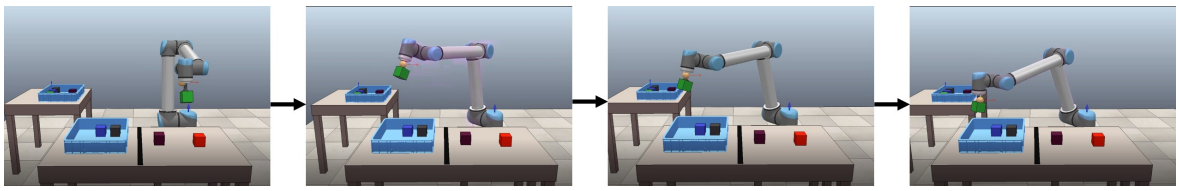
As illustrated in Fig. 4.10, the sub-goal planner can replicate the planning strategy observed in the demonstrations. It also demonstrates its generalizability when presented with different goal positions while ensuring safety between objects. Furthermore, it successfully generates sub-goals for scenarios where the robot needs to pick objects from previously unseen positions, as shown in Fig. 4.11a. However, it's important to note that these positions should remain within a particular range (refer to Section 4.4.3). The primary failure case occurs when the predicted sub-goal is not efficient for grasping the object's surface effectively.

4.4.2.2 Motion Module Performance

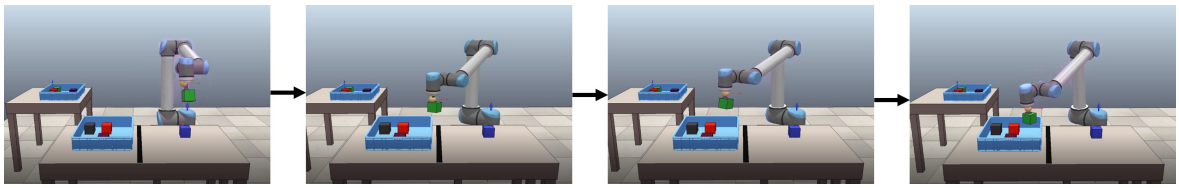
Figure 4.11 illustrates the 3D trajectories generated by the motion module. In the picking stage shown in Figure 4.11a, the motion module imitates the expert's planning strategy, initially positioning itself above the object and subsequently picking it up. As shown in Figure 4.11b, the module effectively handles diverse object positions during the picking stage, producing trajectories that lead to various goal positions, even when starting from the same initial positions in the placing stage.



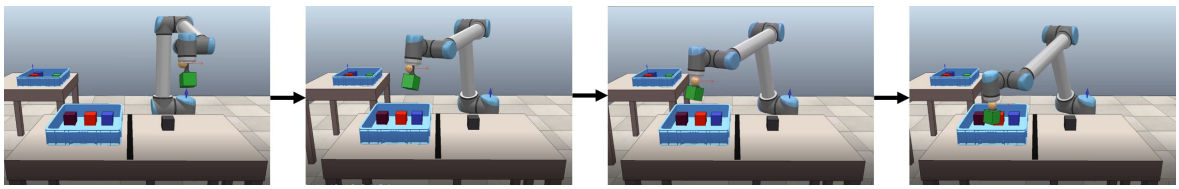
(a) Trained sub-goals from demonstration for g_1



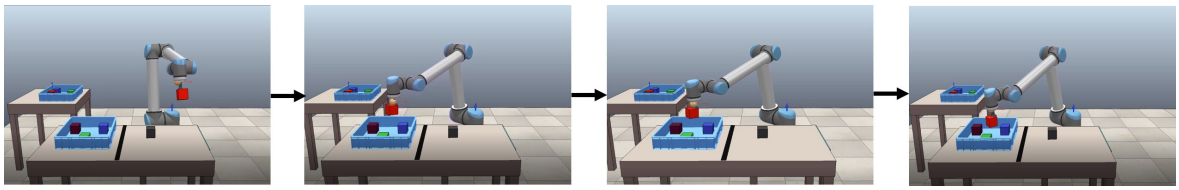
(b) Generalized sub-goals for g_2 with similar observation.



(c) Generalized sub-goals for g_3 with similar observation.

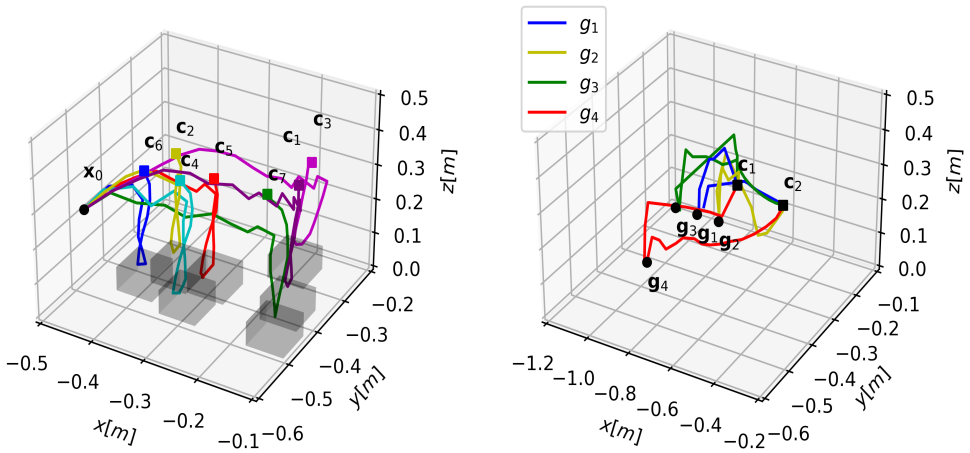


(d) Generalized sub-goals for g_4 with similar observation.



(e) Generalized sub-goals for g_1 with different observation.

Figure 4.10: The simulation results in 4 out of 5. In each figure, the first graph illustrates the initial state of the stage. As shown in the figures, the sub-goal planner exhibits adaptability to different task information while ensuring collision-free motions. Fig.4.10b,4.10c and 4.10d demonstrate its capability of generalizing trajectories based on different task labels while with similar observations. It is also adaptable to different observations with the same task label as shown in Fig.4.10e



(a) 3D representations illustrate the trajectory of the robot end-effector at the picking stage for various object positions. In this representation, x_0 denotes the initial position, and the robot's task is to approach the surface of the objects and reach the picking points marked as c.

(b) 3D representation shows the end-effector's trajectory at the placing stage, starting from various picking points as c and guiding the objects to their predefined goal positions marked as g.

Figure 4.11: The 3D reproduction trajectory is generated by the motion module, which was trained using experiments 4 out of 5.

4.4.3 Overall Performance

In this section, the whole architecture performance in every experiment is discussed. It is compared with different baseline methods including:

1. **DRL**: A Deep Reinforcement Learning(DRL) method uses a flat structure to generate actions directly from full observations. The observations contain objects positions $o = \{o_1, o_2, o_3...o_m\}$ and end-effector positions r. The action space is the same as the neural dynamic planner. For every trial, the goal also contains m objects positions including $g = \{g_1, g_2, ..., g_n\}$ for the selected objects while the rest objects' positions remain the same. DDPG and Hindsight Experience Replay (HER) Buffer is implemented [19] with sparse reward. DRL agent is trained for 5000 iterations for every experiment.

2. **RM+RRT-Connect**[138]: **RRT-Connect** refers to a conventional path planning algorithm that searches a configuration space with two rapidly expanding

random trees growing from both the initial start point and target point. Since it requires both starting and target position, the reasoning module (**RM**) is used to produce the target object's position and its goal. The motion module is replaced by **RRT-Connect**. Moreover, in order to produce collision-free motions, the environmental information is provided in simulation via OMPL library[139].

Regarding variations in object positions within the environment, the proposed system exhibits robustness. It can effectively manage variations of up to 0.3 meters on the x-axis and 0.15 meters on the y-axis concerning object-picking positions. Additionally, it can handle variations of up to 0.35 meters on the x-axis and 0.2 meters on the y-axis for goal positions. The system can accommodate a range of Euclidean distances between objects and their respective goals, spanning from 0.38 meters to 0.82 meters, showcasing its adaptability in different scenarios.

Table 4.3 highlights the challenges faced by the Deep Reinforcement Learning (DRL) algorithm when handling high-dimensional observations and goal spaces, as well as its inefficiency in terms of data. In contrast, the hierarchical structure of the proposed TAMP system proves advantageous, as it efficiently reduces high-dimensional observations into informative features for each module. Moreover, the motion module demonstrates competitive performance in comparison to the collision-free Rapidly-exploring Random Tree Connect (RRT-Connect) algorithm.

Planning and execution times for both the proposed motion module and the **RRT-Connect** algorithm are calculated and compared. In the proposed system, three sub-goals are planned sequentially for each object manipulation, with a maximum of 30 action steps allowed to achieve each sub-goal. During testing, the planning time for the neural network-based planner is approximately 1.3 milliseconds for one of the sub-goals. During manipulation, the joint actions are constrained within a range of -5 to 5 degrees. Consequently, the total action steps for a single object manipulation can vary between 27 and 47 steps, while the execution time can vary between 10.51 seconds and 17.94 seconds.

When comparing the motion module with RRT-Connect, an inverse kinematic solver is employed to iteratively determine optimized joint configurations. Environmental information is provided at the beginning of each object manipulation to enable collision-free motions. In this setup, objects that do not need to be manipulated are considered as obstacles. It is observed that RRT-Connect often requires a longer average planning time (9.6 seconds), and the execution time varies between 18.3 seconds to 25.2 seconds. Additionally, the configuration space becomes more complex as the number of objects increases, impacting the success rate (SR) values. Moreover, the inverse kinematic solver may generate unsafe joint configurations due to kinematic redundancy.

	2 out of 3	3 out of 3	3 out of 5	4 out of 5		
Model	SR	SR	SR	SR	Avg. planning time	Avg. execution time
Ours	0.96±0.012	0.94±0.008	0.92±0.012	0.89±0.017	7.8ms±0.32ms	14.25s ± 2.9s
DRL	0.41±0.032	0.31±0.023	0.18 ±0.018	0.07±0.011	-	-
RM+RRT-Connect	0.94±0.008	0.93±0.012	0.89±0.01	0.88 ±0.012	9.6s ±3.32s	21.75s ± 2.2s

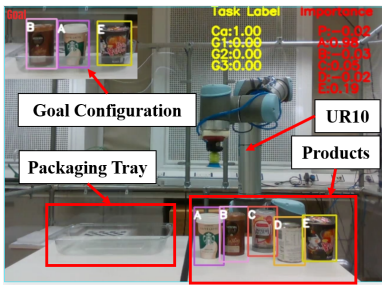
Table 4.3: The overall performance of different methods is compared across two different cases. The DRL agent is trained using demonstrations from both cases, while the TAMP system is only trained on **CASE 1**.

4.4.4 Physical Experiment

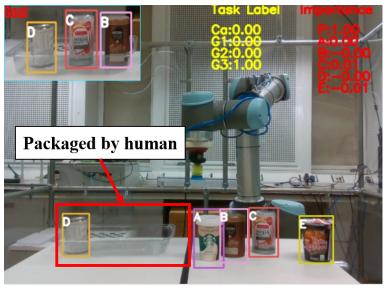
Table 4.4: Physical experiment results in both cases.

	2 out of 3	3 out of 3	3 out of 5
SR in Case 1	0.98±0.002	0.95±0.01	0.93±0.008
SR in Case 2	0.91±0.012	0.91±0.007	0.87±0.034
Avg. time	44.3s±1.45	43.2s±2.3	43.5s±1.8

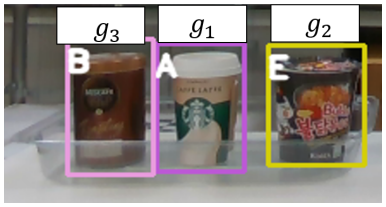
In the physical experiments, a setup similar to the simulations is used. The proposed TAMP architecture can be directly applied to physical experiments as long as the distribution of object positions remains similar to that in the simulations. A RealSense 3D camera is positioned in front of the workspace to sense the environment, as illustrated in Fig.4.12. To obtain real-world information about the products' positions



(a) Case 1:Handling the same task structure



(b) Case 2: Handling different task structure caused by human



(c) Goal configuration with label

Figure 4.12: The physical experiment in 3 out of 5 experiments. The text displayed at the top right corner of the images represents the real-time task plan generated by the reasoning module.

relative to the robot base, a CNN-based detection network YoloV3 [140] is utilized, and coordinate transformation is facilitated through the Robot Operating System (ROS) [141]. The suction cap is controlled by toggling the digital I/O output on the UR10 robot during a single object manipulation cycle. The desired joint actions generated by the motion module are used to control the robot through ROS and MoveIt [142]. One limitation of the neural dynamic planner is its inability to adjust the orientation of the end-effector in real-time. However, object orientations are adjusted using MoveIt during the picking stage, when picking objects with the predicted sub-goal, and in the final step of the placing stage to enhance object detection.

Fig.4.15 provides an example from the 3 out of 5 experiments in **CASE 1**. As described, the TAMP system follows the same learned task structure as in the simulation, progressing from A to B until the final goal configuration is achieved. In **CASE 2**, the worker can initially package the products according to their preference. Subsequently, the robot takes over and begins planning and packaging the remaining selected

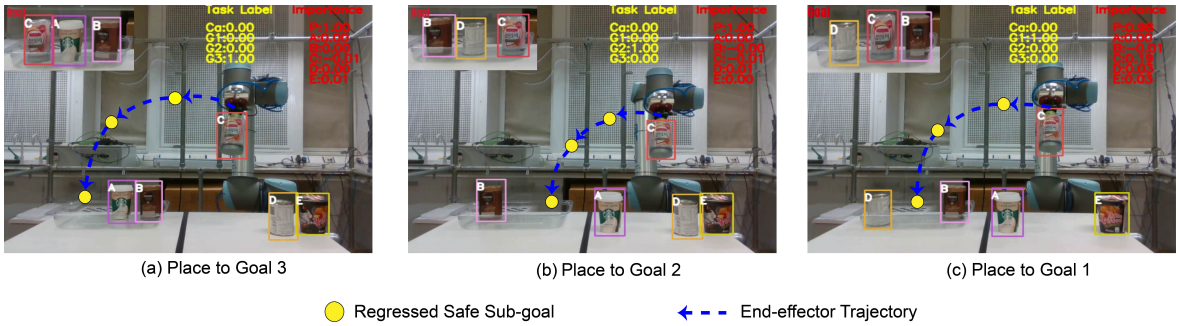


Figure 4.13: The efficacy of the motion module in adapting to different goal positions is demonstrated through physical experiments. For instance, the motion module was initially trained to place object C at Goal 3 in Fig.4.13a. The module showed adaptability by successfully generating trajectories for other untrained task goals, as illustrated in Fig.4.13b and Fig.4.13c.

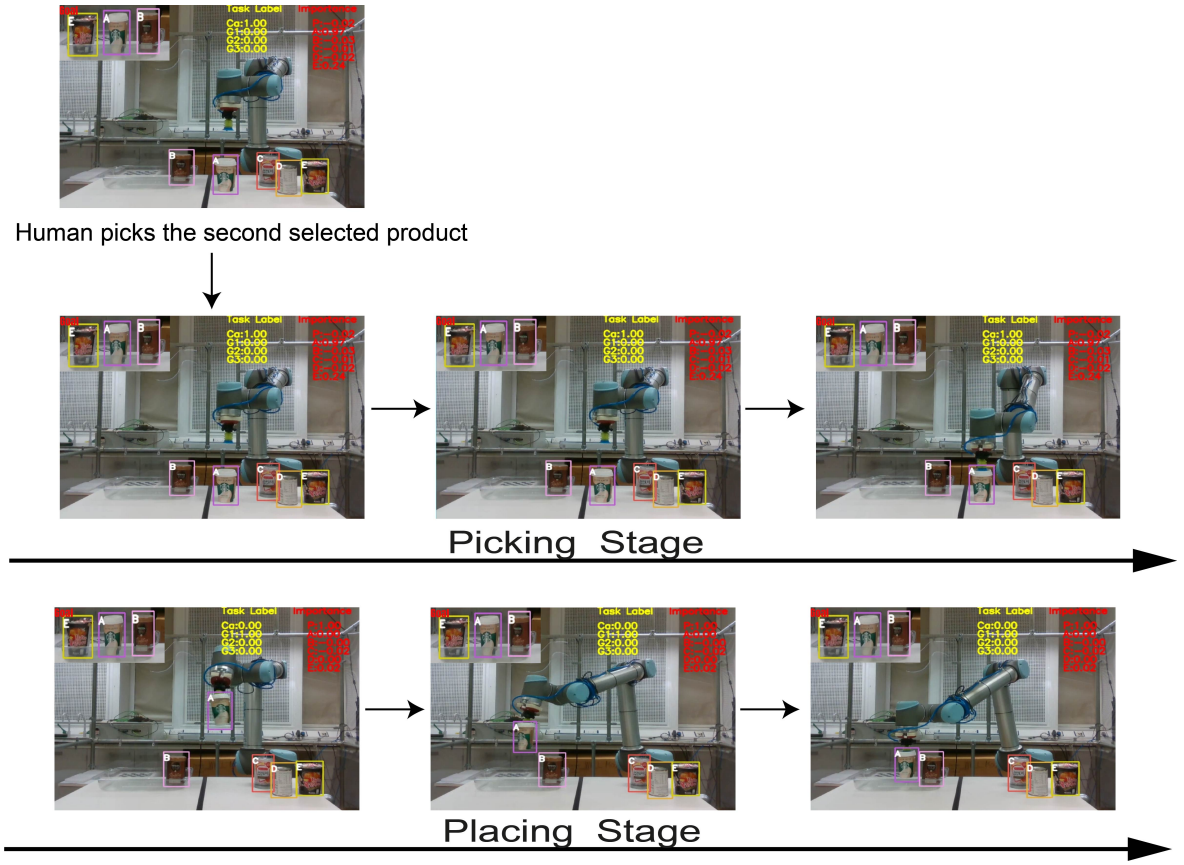
products, as demonstrated in Fig.4.14. The performance of the motion module in the physical experiment is visualized in Fig.4.13.

The experiments were conducted with 20 runs for each case in every experiment, covering scenarios such as 2 out of 3, 3 out of 3, and 3 out of 5. The results in terms of Success Rate (SR) indicate that the proposed system, trained in simulation, can effectively generalize to hardware for both the reasoning module and the motion module, as summarized in Table 4.4. The reasoning module takes approximately 5ms to construct the graph and make predictions, while the planning time for the sub-goal planner is similar to that in the simulation. The Euclidean distance between the object and the goal varies from 0.25m to 0.83m, resulting in a variation in the number of steps taken for each object manipulation, ranging from 20 to 47 steps. The velocity scaling factor for the UR10 robot was set to 0.05 to ensure safety between the robot and humans, resulting in an average time of approximately 0.7s for each step. Additionally, an average of 10.2s is required for extra orientation correction at each stage. It should be noted that RRT-Connect was not implemented due to the need for rich environmental information.

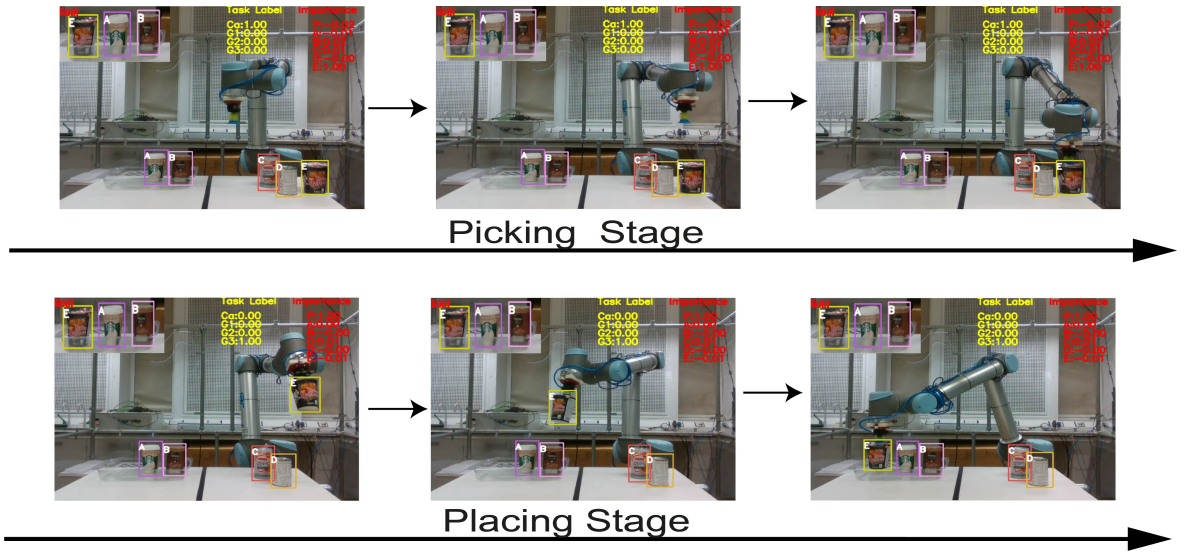
However, it's important to note that the primary reason for failures in the physi-

cal experiments is the unpredictable dynamics of real products during manipulation. These dynamics can lead to unexpected motions during the placing stage, resulting in collisions with other products. Additionally, misclassifications of the products in the YOLOV3 detection system may decrease the success rate during the experiments.

Despite these issues, the proposed architecture demonstrates faster planning and execution times compared to conventional motion planning methods, highlighting its potential for efficient and flexible robotic manipulation in real-world scenarios.

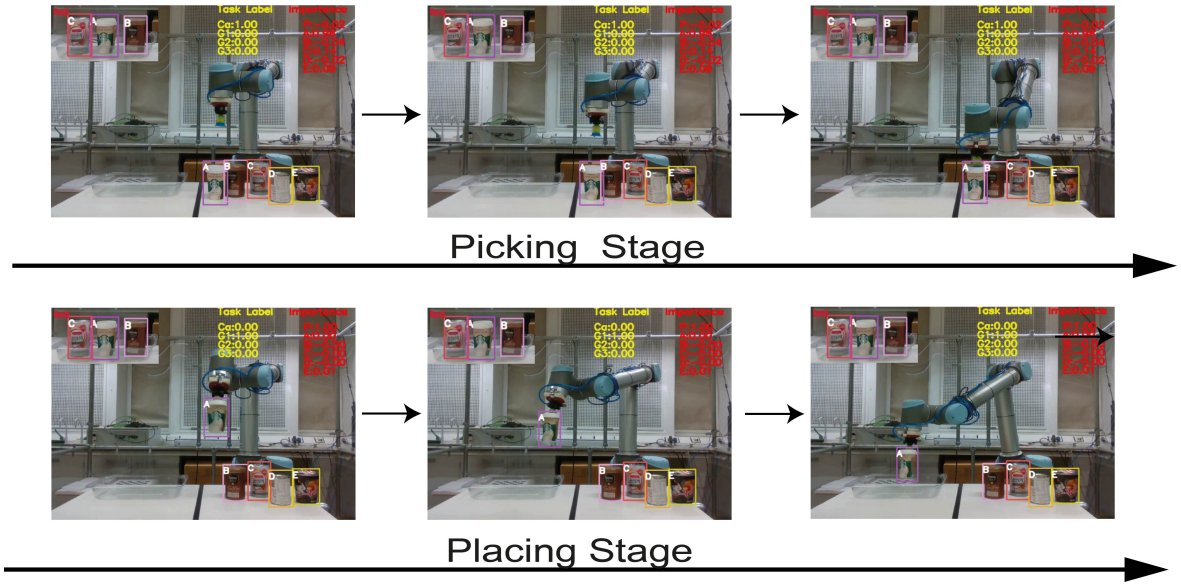


(a) First product manipulation

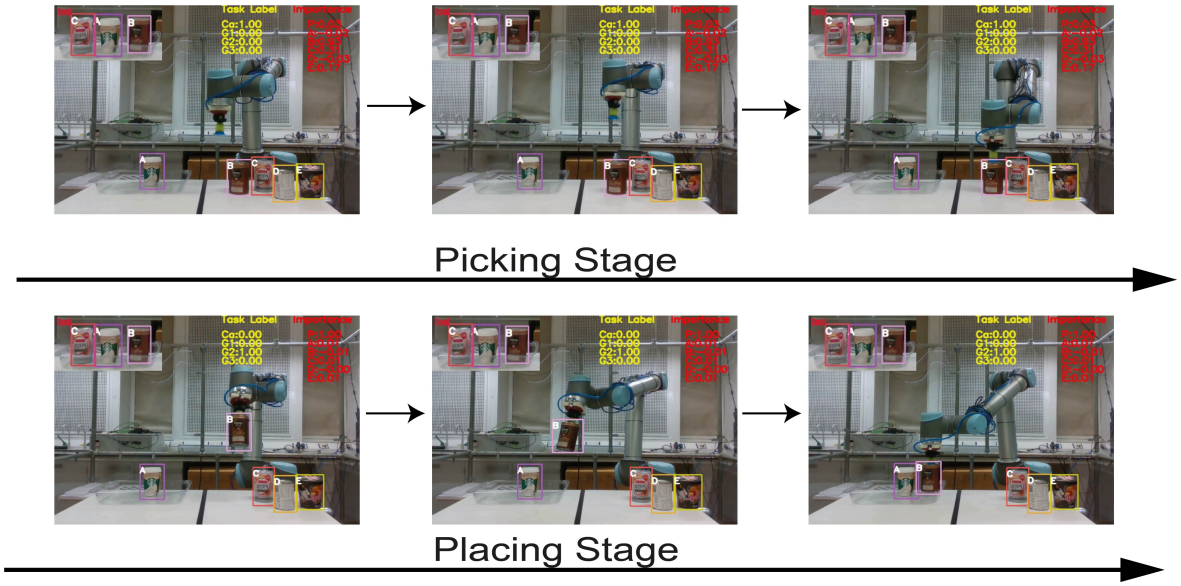


(b) Third product manipulation

Figure 4.14: In the physical experiment in **CASE 2** where 3 out of 5 tasks were conducted, the human participant selected the second object and then left the remaining tasks to the robot.

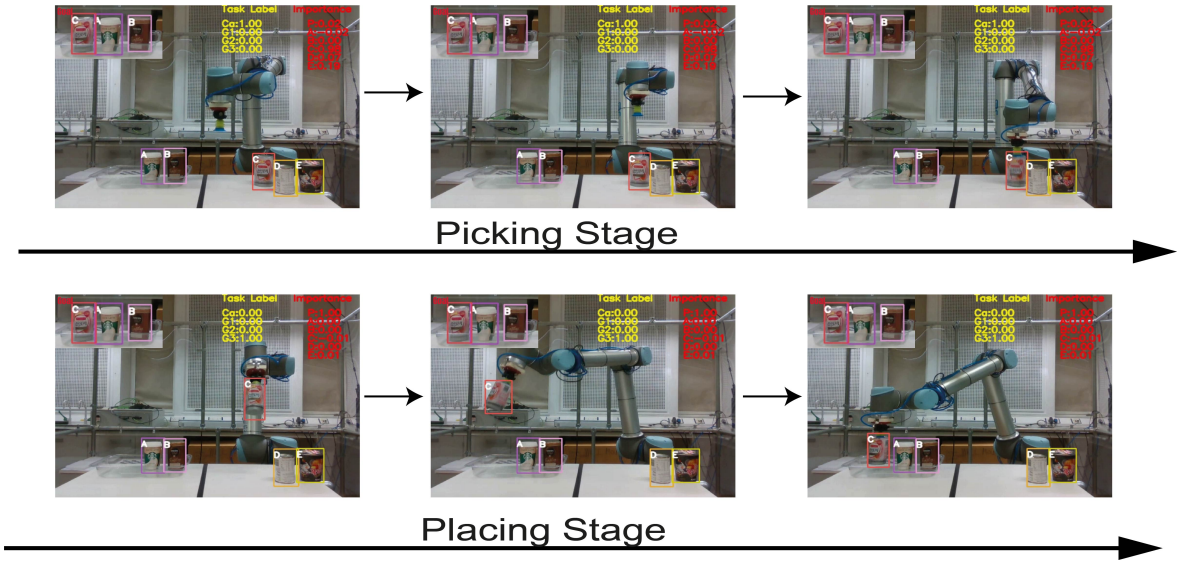


(a) First product manipulation



(b) Second product manipulation

Figure 4.15: In the physical experiment described in **CASE 1**, which involved handling 3 out of 5 tasks, the robot autonomously executed the entire customer order packaging process.



(c) Third product manipulation

Figure 4.15: **(cont.)**.

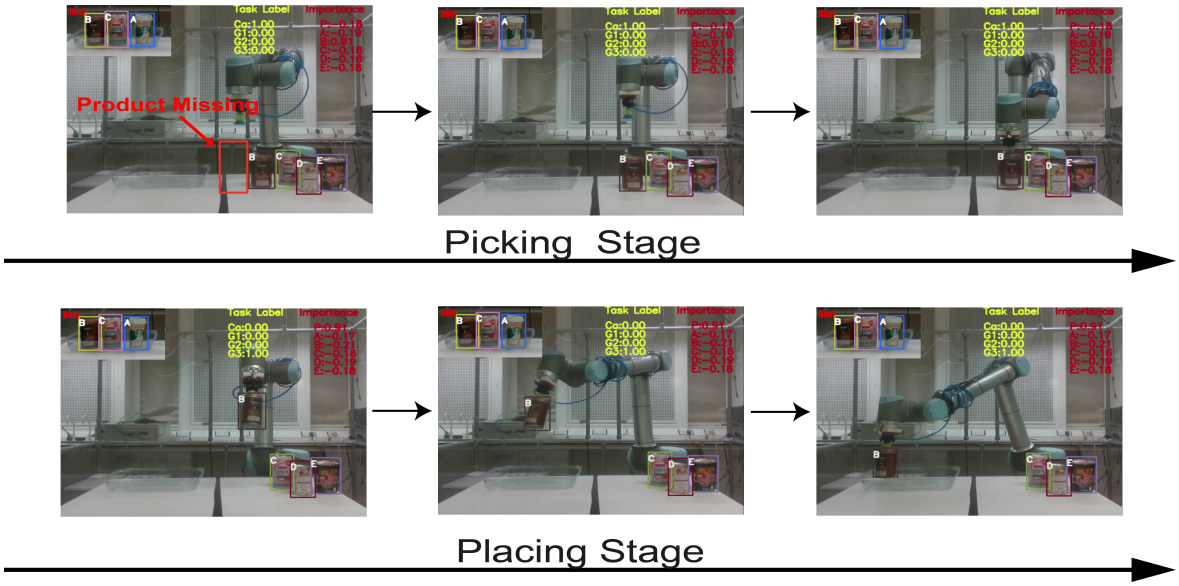
4.4.5 Practical scenarios

To further investigate the capabilities of the proposed framework in handling variations, two practical problems are formulated, which are commonly encountered in mPTO scenarios.

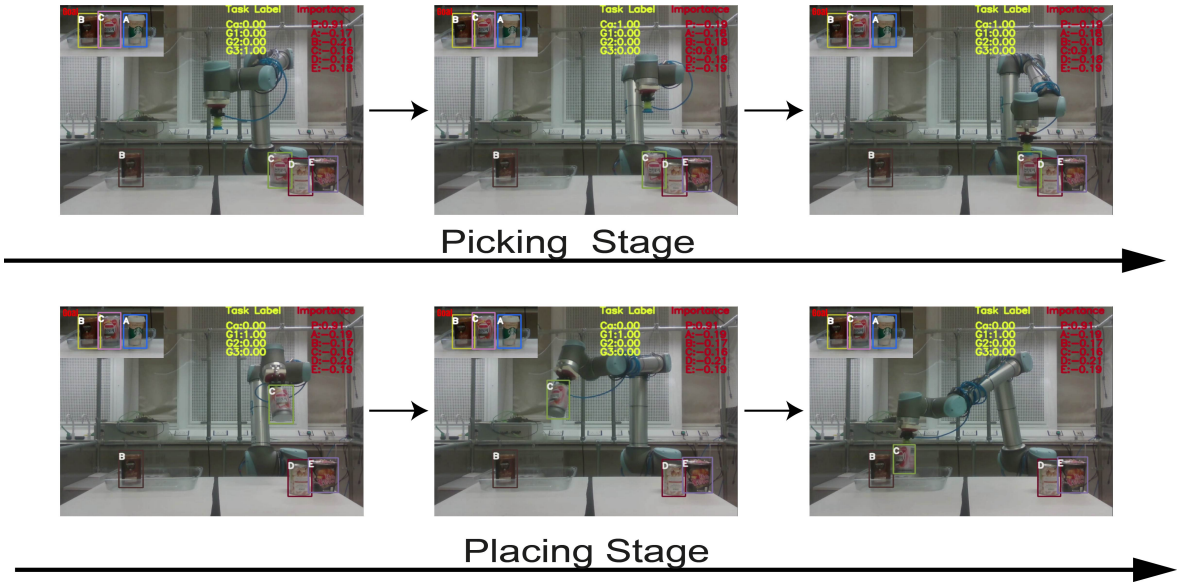
The first practical problem can occur when some items are missing from the inventory and await replenishment. In such a scenario, with the goal of meeting customer order fulfillment requirements, the robot must initially pack the available products into their corresponding destinations while awaiting the arrival of the missing products. Consequently, this situation necessitates a different task structure than the one previously learned. This scenario can be categorized as a partial observability problem, where the dimensions of observations may vary. When using the **MLP** and **RF** approaches for high-level decision-making, they encounter difficulties as they require consistent observation dimensions during both training and testing. Conversely, a GNN-based approach is better suited for handling such a problem, as it can handle varying numbers of nodes. In this context, the reasoning module needs partial re-

training with additional augmented data, consisting of 50 randomly selected partial observable expert demonstrations from 3 out of 5 experiments. This retraining enables the reasoning module to make improved inferences based on each object's position information. During the testing phase, a total of 180 experiments are conducted, with each scenario repeated five times, involving 3 out of 5 scenarios with 1 or 2 products missing. This approach achieves an average SR of 0.97. In addition, 20 physical experiments are conducted with the entire system, resulting in an average SR of 0.92 across five test sets.

The second practical problem arises when customers choose different product combinations, such as **2 out of 5**, **3 out of 5**, and **4 out of 5**. In such cases, the robot must be capable of handling various combinations and varying task lengths simultaneously. The reasoning module is initially trained for the **2 out of 5** scenario using demonstrations from **CASE 1**. The end goal position for each object can be randomly selected, as described in Fig.4.6. Consequently, the system's generalizability is assessed in the context of **3 out of 5** and **4 out of 5** scenarios in both **CASE 1** and **CASE 2**, as illustrated in Fig.4.17. For each experiment, 100 test sets are conducted. Remarkably, the proposed approach, initially trained with a simpler scenario (**2 out of 5**), demonstrates the capability to adapt to more complex and unseen tasks, such as **3 out of 5** and **4 out of 5**.

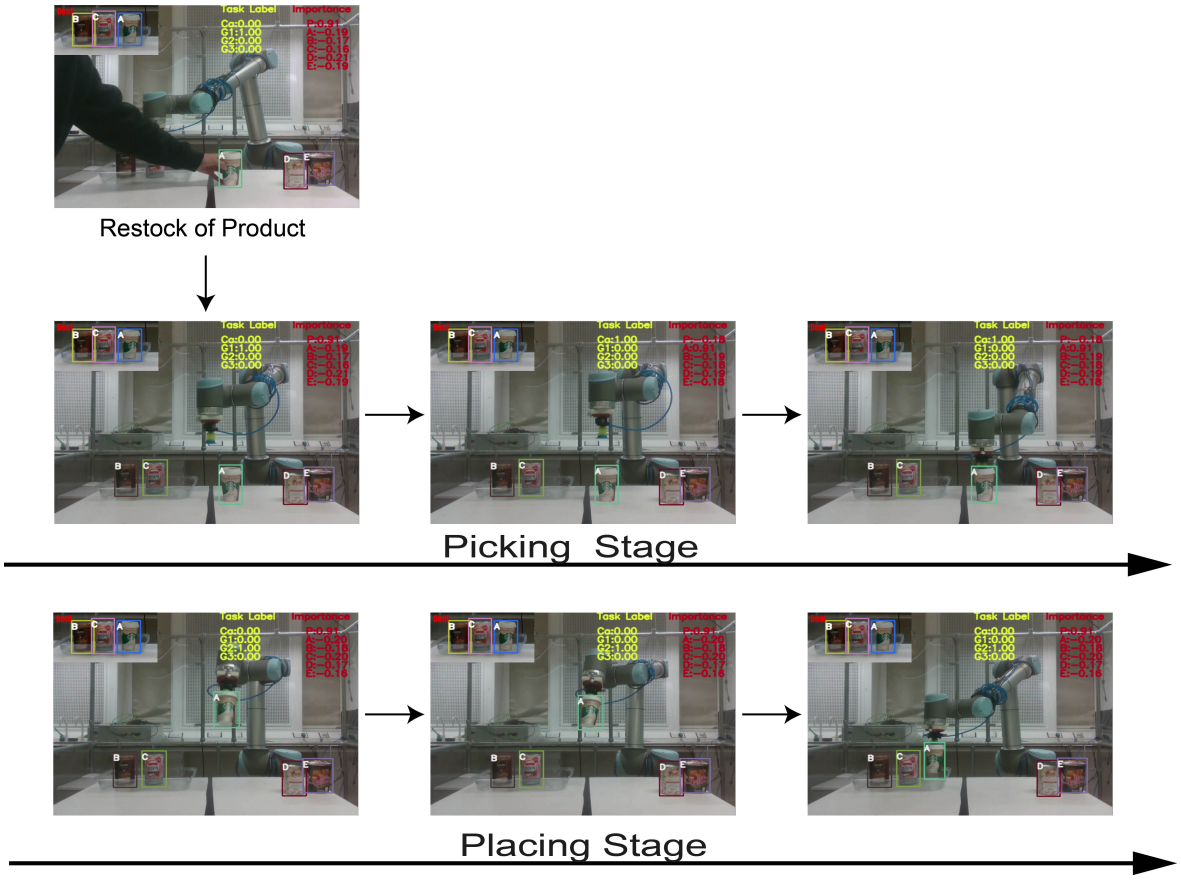


(a) Second product manipulation



(b) Third product manipulation

Figure 4.16: Physical experiment for 3 out of 5 in **CASE 1**, where some products are out of stock while robot needs to firstly pack the rest and come back to pack the restocked product.



(c) First product manipulation

Figure 4.16: Physical experiment for 3 out of 5 in **CASE 1**, where some products are out of stock while robot needs to firstly pack the rest and come back to pack the restocked product. **(cont.)**.

4.5 Summary

In this section, the proposed TAMP architecture is discussed while revisiting the research questions raised in section 4.1 based on the results from the experiments conducted.

In the context of high-level decision-making, the reasoning module demonstrates the capability to directly learn the fundamental task structure from demonstrations. This process is achieved without the necessity for predefined task structures or additional efforts, such as the use of specific *if-else* rules. This approach effectively addresses the

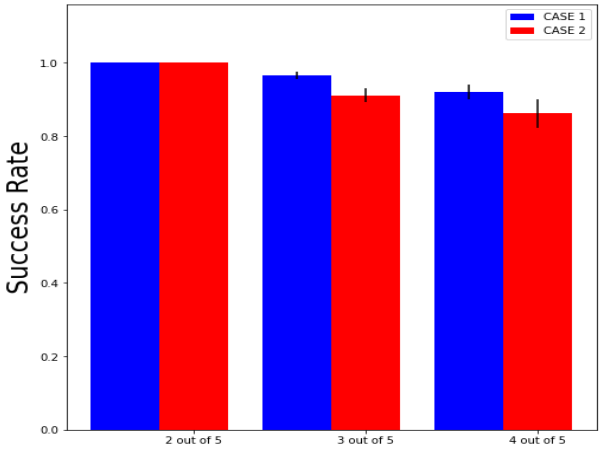


Figure 4.17: A practical problem arises when customers have the flexibility to select varying numbers of products. The TAMP system is initially trained with a scenario where customers choose "2 out of 5" products. It demonstrates the ability to generalize to scenarios where customers select "3 out of 5" products with an average success rate (SR) of 0.97 in CASE 1 and 0.91 in CASE 2. Moreover, it also generalizes to scenarios where customers choose "4 out of 5" products, achieving an average SR of 0.92 in CASE 1 and 0.86 in CASE 2.

concern raised in **Q1**

In response to **Q2**, the introduced framework adeptly converts variations in the high-level plan into distinct low-level motion executions. This is achieved by dividing a single object manipulation task into discrete stages of picking and placing. Such segmentation enables the reasoning module to rank the specific features necessary for the motion module at each stage of manipulation. Additionally, the classified labels P^g_{pred} are utilized to assist the motion module in distinguishing between various sub-tasks. Moreover, this motion module is designed to not only effectively learn from an expert's planning strategy but also to adapt to diverse observational data.

To tackle both **Q1** and **Q3**, the proposed framework is equipped to manage redundant objects observed in the process by incorporating an additional property, denoted as I, within the Graph Neural Network (GNN). This inclusion allows the GNN to concentrate on a specific subset of products during a mPTO task. Moreover, the results underscore the framework's capability for zero-shot generalization, particularly in adapting to various task structures that arise due to differences in human performance,

as elaborated in Fig.4.9.

In regard to the limitations of this work, a comparison with [74] indicates that their approach has an advantage in terms of generalizability to diverse geometric shapes of final goals. However, the current study is limited to using pose-specified labels to enable the functionality of the low-level motion module. On a different note, the reasoning module in this study is adept at handling various task structures, making it potentially more suitable for accommodating the diverse and changing preferences found in Human-Robot Collaboration scenarios.

An additional limitation of the motion module lies in the sub-goal planner's ability to manage substantial variations in observations. These observations are characterized as multivariate Gaussian distributions, which may result in inefficient planning, particularly in accurately picking an object at its surface. Furthermore, the neural dynamic planner in this framework is limited to considering only the 3D positions of the robot's end-effector. While this may be sufficient for interactions with rigid objects, the significance of this limitation becomes more pronounced when dealing with semi-rigid or soft objects.

Chapter 5

A vision-based adaptive task planning framework for varying Human-Robot-Collaboration

5.1 Introduction

This chapter proposed a vision-based task planning system through LfD for responsive HRC. In the previous works, the human state is reflected by the object position information and the final end goal is predefined. In this Chapter, the robot has no prior knowledge of the human worker's intended goal. Instead, it should understand human actions and thus infer the varying goals without object pose information. Consequently, it should generate adaptive plans in response to human actions. It is studied as an Assemble-To-Order problem in the manufacturing setting.

Similar to the Package-to-Order (PTO) process mentioned earlier, Assembly-to-Order (ATO) also refers to a production strategy that involves customizing products using various basic components. In the manual ATO process, a human worker retrieves different parts from the inventory and assembles the final products according to customer demands.

Conventional Human-Robot Collaboration (HRC) methodologies typically involve offline programming, necessitating that human collaborators adhere to a predetermined workflow [9]. In such frameworks, robots are capable of providing support through actions like object handling. However, this approach becomes problematic when there are deviations in the workflow, whether due to human variability or changes in the task itself. To address this issue, recent advancements in Learning from Demonstration (LfD) have demonstrated significant promise in facilitating a more adaptable HRC environment. LfD empowers robots to assimilate task structures as demonstrated by humans and to make decisions cognitively in response to varying human actions [9].

To achieve this, understanding human actions becomes a primary task for the robot in Human-Robot Collaboration (HRC). Earlier research has focused on interpreting human assembly scenes (including human poses and the surrounding environment) from spatial images using CNNs [92], [47]. The analysis of human movement trajectories is also critical for robots to recognize assembly intentions. Previous works have utilized Hidden Markov Models (HMM) and Multilayer Perceptrons (MLP) with skeleton joints [96], alongside Recurrent Neural Networks [7] and semi-flexible Neural Networks [97]. Moreover, a combination of temporal motion features with spatial assembly context, as suggested by [98], could more effectively distinguish different assembly actions. Most studies often employ CNNs for object detection and RNNs to identify assembly actions from either depth image sequences [99] or skeleton data [61]. In the context of Assemble-to-Order (ATO) environments, where diverse products are created, different components may be placed in the same position based on customer demands, as illustrated in Fig.5.4. Here, while the assembly motions (moving to a position) remain consistent, the corresponding objects might differ.

The robot is thus tasked with identifying the human's intended goal based on the sequences of human actions observed. Probabilistic methods are frequently employed to infer the likely goals intended by humans. For example, Bayesian Inference has been applied to deduce human navigational goals [105] and assembly plans, incorporating prior knowledge about human poses and object interactions [7]. Additionally, the

variable-length Markov Model (VMM) has been utilized to analyze classified action sequences, thereby aiding in generating optimal plan predictions [47]. These methods, however, often depend significantly on prior expert knowledge.

Additionally, to facilitate robots in Human-Robot Collaboration (HRC) scenarios in assisting human coworkers with planned actions, the concept of hierarchy within demonstrated task structures has been investigated. These task structures can be established using a predefined AND/OR graph, taking into account all possible plan combinations, as discussed in [63], [61], and [62]. The application of Hidden Markov Models (HMM) is also prevalent for identifying the hidden states linking sub-tasks [65]. In the study by [9], inverse reinforcement learning (IRL) was employed to determine the most desirable actions that yield optimal rewards for custom-made products. Hierarchical Task Networks (HTN) have been utilized as well, to define task structures and model transition probabilities, as seen in [66] and [67]. However, these methodologies often necessitate design based on domain-specific knowledge.

For the above discussion, this chapter proposes three common variations that should be addressed in HRC scenarios for the ATO problem:

- **Q1**: Variations in human action sequences: In the context of product assembly, there exists a notable variability in the operational sequences employed by human workers. These variances are attributable to individual preferences, leading to diverse approaches to achieving the completion of various products.

- **Q2**: Variations in goal configurations: This concept pertains to the customization, as described in Fig.5.4. Diverging from previous studies that concentrate on uniform end-products, this study aims to infer the differing end-goals from variable human operational sequences within the Assemble-to-Order (ATO) settings.

- **Q3**: Variations in robot task planning: This is about the strategies robots use to help at different stages of assembly. Previous research allowed robots to make simple plans like "handle an object" based on inferred goals. This study, however,

is about making robots plan more detailed actions like "handling an object in multiple positions."

To this end, a two-level Learning from Demonstration (LfD) framework is proposed based on only 2D videos. This framework can be described as:

- Firstly, a hand-centric action detector is designed to actively identify human assembly actions by integrating temporal hand motions and the spatial hand-object interaction status. In this Chapter, the term "actions" in the context of the ATO problem pertains to the identification of which object has been assembled and its specific location by human workers.

- For robot task planning, a semantic planner has been developed that utilizes the detected action sequences to automatically create a product assembly graph. This planner initially employs a Graph Neural Network (GNN) [133] to infer the goal configurations intended by the human worker

- Furthermore, the semantic means that the planner can extract the relationship between objects and the assembly positions through the trained graph nodes' embedding. Consequently, it generates assistive plans as textual descriptions, detailing **which** object should be assembled **where**, based on the current sequence of human actions. Furthermore, this approach is capable of generating plans for new, previously unseen human action sequences."

The proposed framework is validated through physical experiments, specifically focusing on real-world scenarios involving the adaptive assembly of valve brackets. The outcomes of these tests demonstrated the framework's efficiency in accurately identifying assembly actions. Additionally, it was effective in offering detailed guidance and support to collaborators, especially in situations involving diverse end goals

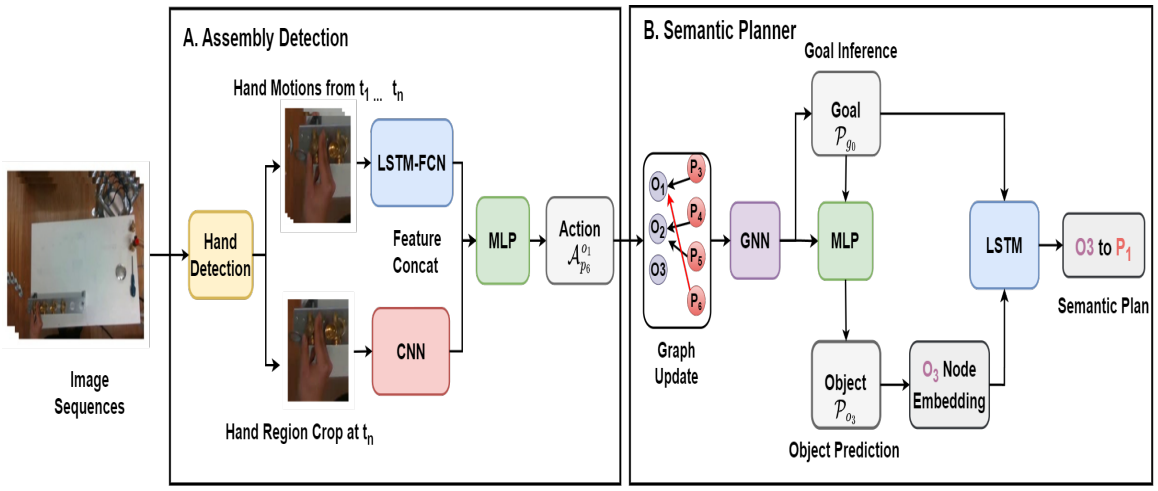


Figure 5.1: Pipeline of the proposed framework:A: The first step involves detecting right hands in incoming image sequences using MediaPipe. The framework then classifies the assembly actions by combining hand region features extracted through a CNN-based extractor with motion features obtained from LSTM-FCN. B: The classified action is used to update the assembly graph, serving as a connecting edge. The framework, leveraging this updated graph, adaptively infers the human's intended goal configuration and determines the next object to be assembled using GNN. Subsequently, a simple LSTM translates the object node's embedding into a language format, indicating its preferred assembly positions. This language output is useful for guiding human decisions in future steps or for directing robot actions to assist the co-worker.

5.2 Methodology

Assume a Assemble-To-Order (ATO) scenario for one end-product with z goal configurations $G = \{g_i\}_{i=1}^{i=z}$ according to customer demands. It is assumed that there are n **types** of components or objects $O = \{o_i\}_{i=1}^{i=n}$ and that each of them has different amounts. Moreover, assume there will be m possible assemble positions $P = \{p_m\}_{m=1}^{m}$ for O towards achieving final goal configurations.

The framework, as illustrated in Fig.5.1, is designed to establish a Human-Robot Collaboration (HRC) system that is capable of dynamically detecting human actions. The primary objective of this system is to accurately identify the human's intended goal, denoted as g. Based on this recognition, the system will then generate a detailed plan for the robot. This plan specifies the position, represented as p, where the next

workpiece, o, should be assembled. It is important to note that the position p may vary for each workpiece o.

5.2.1 Hand-centric Action Detector

The objective is to characterize assembly actions by identifying **which** object has been assembled and **where** it has been placed. To achieve this, a hand-centric action detector is introduced. This approach is based on the premise that hand regions in a scene are often the most informative compared to other features. The open-source platform MediaPipe [143] is initially employed to capture hand crops and to use segments of hand trajectories for extracting pertinent temporal features.

More specifically, the input vector at each frame t is the normalized hand position $s_t = \{x_t, y_t\}$ in the image space and LSTM Fully Convolutional Neural Network (LSTM-FCN) is used to [144] to extract temporal features. Compared to a single LSTM and temporal convolution layer, this model requires less preprocessing and is more robust to noise. This is achieved by combining the character of time series dependency from LSTM and time-invariant features from FCN [144]. In detail, the LSTM block will produce the hidden state h_t at the last step as $h_t = LSTM(h_{t-1}, s_t)$. The 1D convolution layers $conv1d$ extract features as $f_{FCN} = \mathbf{Avg}(\sigma(conv1d(s)))$, where σ is the ReLu activation function, **Avg** is the global average polling function and s is the same time-series input as LSTM. Afterwards, we concatenate these two features to obtain the final temporal features:

$$f_{temporal} = concat(h_t, f_{FCN}) \tag{5.1}$$

Instead of object detection, the spatial features are termed as the interaction status between the hand and objects as shown in Fig.5.2. As the labelling process of object detection can be time-consuming and it can cause errors when hands overlap with objects [99] especially when the objects are small. Thus, a CNN-based VGG16 [145] is adopted as the backbone feature extractor and the extracted feature can be expressed

as Eq.5.2, where I is the hand-centric crop at the last frame.

$$f_{spatial} = CNN(I) \tag{5.2}$$

Finally, these two features are fused into a Multilayer perceptron (MLP). From this, the classified assemble actions can obtained as $\mathcal{A} = \{\mathcal{A}_{p_1}^{o_1}, ..., \mathcal{A}_{p_m}^{o_n}\}$, and $\{\mathcal{A}_{trans}, \mathcal{A}_{screw}\}$, where $\{\mathcal{A}_{p_1}^{o_1}, ..., \mathcal{A}_{p_m}^{o_n}\}$ represents the probability of the object o_n has been assembled to p_m with examples presented in Fig.5.2a,5.2b and 5.2c. These classified actions will be further used in the semantic planner. There are also intermediate actions such as fetches and transports as $\mathcal{A}_{trans}$ as shown in Fig.5.2d and 5.2e. An additional $\mathcal{A}_{screw}$ is used to recognise the screwing intention as Fig.5.2f and thus to trigger the future robot actions. Given the $\hat{\mathcal{A}}$ as the ground truths, the optimal assemble actions can be obtained by training the proposed action detector in an end-to-end fashion with cross-entropy loss as Eq.5.3, where θ are the trainable parameters.

$$\mathcal{L}_a = argmin_\theta[-\sum_{i=1}^{i}(\hat{\mathcal{A}}_i log\mathcal{A}_i + (1-\hat{\mathcal{A}}_i)log(1-\mathcal{A}_i))] \tag{5.3}$$

(a) Assemble with o_1 (b) Assemble with o_2 (c) Assemble with o_3

(d) Fetch object (e) Transport with o_2 (f) Interaction with screw

Figure 5.2: Different examples of spatial assembly actions

5.2.2 Graph-based semantic planning

The purpose of the semantic planner is to learn **which** object should be handled to **where** based on inferred **varying** goals configurations.

The assembly scenes are encoded as assembly graph $\mathcal{G}$ containing $n+m$ nodes as $V = \{v_{o_1}, .., v_{o_n}, v_{p_1}, ..., v_{p_m}\}$, where v_{o_n} is the object node and v_{p_m} is the position node. Each node contains two-dimensional categorical features with one describing the type of the objects and the other one describing the assembly position information. The detected assembly actions $\{\mathcal{A}_{p_1}^{o_1}, ...\mathcal{A}_{p_m}^{o_n}\}$ are thus used to build the adjacency matrix $E = \{e_{p_1}^{o_1}, ..., e_{p_m}^{o_n}\}$, where $e_{p_n}^{o_n}$ is a directed linking edge to describe the connectivity between v_{o_n} and v_{p_m}. More specifically, the neighbour nodes of an object node v_{o_n} will be the position nodes v_{p_m} according to the classified actions:

$$e_{p_m}^{o_n} = \begin{cases} 1 & \mathcal{A}_{p_m}^{o_n} = 1 \\ 0 & otherwise \end{cases} \tag{5.4}$$

Afterwards, a GraphSAGE(Sage) [133] layer is adopted in this work to process the graph observations. It aggregates the mean features embedding from the neighbour nodes of each object node as $\Gamma_{\mathcal{N}(i)}^k = \frac{1}{N}\sum_{j\in\mathcal{N}(i)}(\Gamma_j^{k-1}, j \in \mathcal{N}(i))$, where k stands for the kth Sage layer. This aggregated embedding will be concatenated with target node embedding at the previous layer and further multiplied by a weight matrix W_k as Eq.5.5, where $f_{\theta_{gnn_{k-1}}}$ and $f_{\theta_{gnn_k}}$ are the Sage layers with trainable parameters $\theta_{gnn_{k-1}}$ and θ_{gnn_k}, σ is the ReLu activation function.

$$\Gamma_i^k = \sigma(W_k \cdot (f_{\theta_{gnn_{k-1}}}(\Gamma_i^{k-1}) + f_{\theta_{gnn_k}}(\Gamma_{\mathcal{N}(i)}^k)) \tag{5.5}$$

A readout layer is thus used to aggregate nodes embedding Γ_i^k into graph embedding as Eq.5.6. A final output layer will accept Φ_k and generate z dimensional outputs

$\mathcal{P}_g^{pred} = \{\mathcal{P}_{g_1}^{pred}, ..., \mathcal{P}_{g_z}^{pred}\}$ which describe the inferred goal configuration g_z.

$$\Phi_k = \frac{1}{\mathcal{N}(v)} \sum_{i \in \mathcal{N}(v)} \Gamma_i^k \tag{5.6}$$

The semantic planner further predicts the next object should be assembled as $\mathcal{P}_o^{pred} = \{\mathcal{P}_{o_1}^{pred}, ..., \mathcal{P}_{o_n}^{pred}\}$ through another MLP with the inputs containing graph embedding Φ_k and inferred goal $\mathcal{P}_g^{pred}$ as:

$$\mathcal{P}_o^{pred} = MLP(\Phi_k, \mathcal{P}_g^{pred}) \tag{5.7}$$

To train this model, it is considered as a classification problem with the demonstrated ground truth $\mathcal{P}_g^{tgt}$ and $\mathcal{P}_o^{tgt}$. $\mathcal{P}_g^{pred}$ and $\mathcal{P}_o^{pred}$ are jointly learnt via cross-entropy loss:

$$\mathcal{L}_g = argmin_\theta[-\sum_{n=1}^{n}[\mathcal{P}_{o_n}^{tgt}]log(\mathcal{P}_{o_n}^{pred}) - \sum_{z=1}^{z}[\mathcal{P}_{g_z}^{tgt}]log(\mathcal{P}_{g_z}^{pred})] \tag{5.8}$$

In ATO scenarios, a simple multi-class classification is inefficient in producing a plan such as "handling objects to multiple positions". As it will always produce a deterministic result (i.e. the label with the highest probability). An advantage of the graph construction in this work is that: for each object node, it only aggregates the assembled position information that is relevant to itself at different HRC stages under different goal configurations. Therefore, this work aims to produce "semantic plans" that interpret the objects' graphical observations through a simple LSTM with the inferred goal.

During training, the LSTM model commences with the input features including the node embedding Γ_o^k and the label $\mathcal{P}_g^{pred}$ denoted as $f_{lstm} = \{\Gamma_o^k, \mathcal{P}_g^{pred}\}$. Concurrently, the ground truth captions (i.e. semantic plan) are processed through an embedding layer, converting discrete word indices into continuous vectors:

$$\textbf{embeds} = W_{\text{embedding}}[\text{captions}] \tag{5.9}$$

where $W_{\text{embedding}}$ represents the embedding matrix. These embeddings, denoted as **embeds**, are then concatenated with the input features to form the complete input for the LSTM layer as shown in Eq.5.10. This is known as the "teacher forcing" strategy in order to improve the training and enhance model stability.

$$\textbf{inputs} = concat(f_{lstm}, \textbf{embeds}) \tag{5.10}$$

The LSTM processes this concatenated input iteratively to learn temporal dependencies. At each time step t, the LSTM updates its hidden state h_t and cell state c_t:

$$(h_t, c_t) = \text{LSTM}(\textbf{inputs}, h_{t-1}, c_{t-1}) \tag{5.11}$$

where h_t captures the short-term dependencies, effectively acting as the memory of recent inputs, while c_t retains long-term information, allowing the LSTM to remember or forget information over extended sequences.

Finally, the output is passed through a fully connected layer to predict the next word with the highest probability:

$$y_t = W_{fc} h_t + b_{fc} \tag{5.12}$$

Here, W_{fc} is the weight matrix of the fully connected layer, and b_{fc} denotes the bias.

During the testing stage, the procedure adopts a slightly different approach. It commences with the f_{lstm} as the initial input to the LSTM. At each time step, the LSTM's output is passed through the fully connected layer, which predicts the most probable word. This word's embedding then serves as the input for the next time step, creating an iterative loop until a termination condition is met. Therefore, it can be simply expressed as:

$$txt = LSTM(\Gamma_o^k, \mathcal{P}_g^{pred}) \tag{5.13}$$

The generated txt will contain the information regarding the object type and its unfinished assembly positions, for example, "o_1, p_3, p_4" refers to o_1 should be assembled to p_3 and p_4 afterwards. Moreover, the planner can recognise the situation when all the positions of o_1 have been completed and generate text as "o_1, Finished".

The proposed semantic planner can thus be used in two ways:

1. Firstly, it can produce txt for each object by feeding objects node embedding into LSTM. This can assist the human worker in future assembly decisions, which is referred to **Semantic Guidance**.

2. Secondly, when robots are involved, the planner will propose the next object that should be handled given the current graph and thus produce **Semantic Control** command, for example, $[o_{next}, p_2, p_3] = LSTM(\Gamma^k_{o_{next}}, \mathcal{P}^{pred}_g)$, to control the robot via a known motion controller such as $PickandPlace(o_{next}, p_2)$.

5.2.3 System Integration

Finally, the complete system is outlined in Algorithm.3 and is designed to operate in real time. To prevent unnecessary updates to the graph $\mathcal{G}$, the human worker is advised to move their hand out of the detection region after completing an assembly task. Consequently, the most recently identified action label $\mathcal{A}_t$, which could range from $\mathcal{A}^{o_1}_{p_1}$ to $\mathcal{A}^{o_n}_{p_m}$, is utilized to update the graph edge, as detailed in the previous section. Following this, the system generates **Semantic Guidance** specific to each object type o.

In the context of Human-Robot Collaboration (HRC), the system is programmed such that when $\mathcal{A}_{screw}$ is detected, the robot will execute a predefined $PickandPlace()$ function to position the object in the initially suggested location by **Semantic Control**. During this process, the human worker can focus on screwing the assembled objects or assembling the object currently managed by the robot. The detector $\mathcal{M}$ is temporarily disabled in these phases, and the graph is updated accordingly. This process continues until each object o is marked as 'Finished'.

Algorithm 3 Proposed HRC system

Initialize assembly graph $\mathcal{G}$

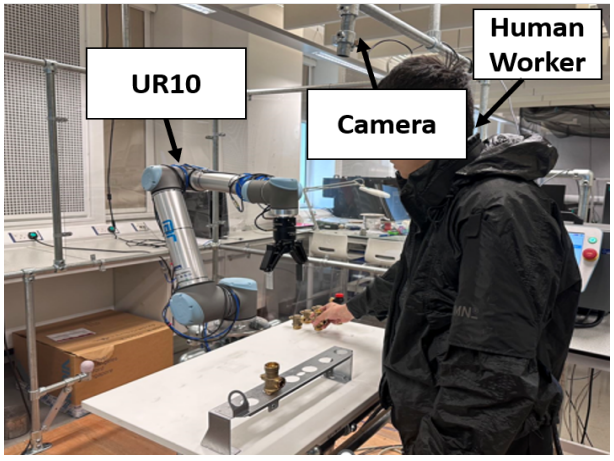
Trained action detection system $\mathcal{M}$, semantic planner $\mathcal{C}$ including a GNN encoder GNN and a decoder $LSTM$

while *Assembly not finished* **do**
- Track hand motion via MediaPipe
- **if** *hand detected* **then**
  - Hand trajectories segment s with t frame length and hand-centric image crop I at the last step
  - Predict assembly actions $\mathcal{A}_t = \mathcal{M}(I, s)$
  - **if** $\mathcal{A}_t \in \{\mathcal{A}^{o_1}_{p_1}, ...\mathcal{A}^{o_n}_{p_m}\}$ ***and*** *hand motion finished* **then**
    - Update Graph Edge $e^{o_n}_{p_m}$ according to assembly actions $\mathcal{A}^{o_n}_{p_m}$
    - Update objects status in pending area
    - Infer final goal $g_z = GNN(\mathcal{G})$
    - Generate **Semantic Guidance** for each object o_n $[o_n, p_1, ..., p_m] = LSTM(\Gamma_{o_n}, g_z)$
  - **else if** $\mathcal{A}_t = \mathcal{A}_{screw}$ **then**
    - Infer final goal according to current graph $\mathcal{G}$
    - as $g_z = GNN(\mathcal{G})$
    - Predict next object $o_{next} = MLP(\Phi, g_z)$
    - Produce **Semantic Control** command $[o_{next}, p_1, ..., p_m] = LSTM(\Gamma_{o_{next}}, g_z)$
    - Robot execution $PickandPlace(o_{next}, p_1)$
    - Update Graph

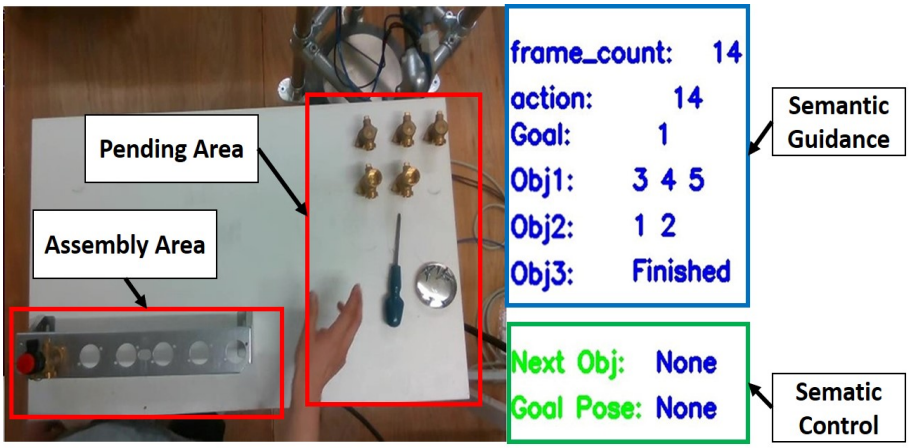
5.3 Experimental Setup

In this section, the effectiveness of the proposed system is demonstrated through a real-industrial ATO scenario. This particular ATO scenario involves assembling valve

brackets, which vary depending on the boiler model. The experiments are structured to validate the efficiency of the Hand-centric Action Detector in recognizing human assembly actions using only 2D video demonstrations. Additionally, the experiments highlight the proficiency of the Semantic Planner in managing diverse goal configurations. This includes its adaptability to different human action sequences and task-planning strategies. Lastly, the overall system's performance in real-time experimental settings is evaluated and presented.



(a) Experimental setup



(b) The designed user interface.

Figure 5.3: The experimental setup and designed user interface

The studied use-case contains three different types of objects $O = \{o_1, o_2, o_3\}$ as shown in Fig.5.2a,5.2b and 5.2c. The number of each type of object varies: the final

(a) Assemble Goal 0

(b) Assemble Goal 1

(c) Assemble Goal 2

Figure 5.4: Different customized goal configurations.

product needs three o_1, two o_2 and one o_3. They can be flexibly assembled to form three different configurations $G = \{g_0.g_1, g_2\}$ as shown in Fig.5.4.

Additionally, a user interface has been developed, as depicted in Fig.5.3b. The assembly cell is monitored from above using a RealSense camera (D435i). Within this setup, the human worker can pick up objects or a screwdriver from the designated pending area, and then proceed to assemble or screw these objects onto a steel bracket in the assembly area. The user interface includes a blue box that displays the count of detected hand frames, the classified assembly actions, the inferred goals, and the remaining assembly positions for each object o, collectively referred to as the **Semantic Guidance**. The green box in the interface presents the **Semantic Control** commands for the robot. The position output, denoted by numbers such as "1,2,3,4,5", indicates the assembly positions from right to left. The robot in the system, a Universal 10 (UR10) arm, is operated via the **Semantic Control** command, using the Robotic

Label	Action	Label	Action	Label	Action	Label	Action
0	o_1 to p_1	5	o_1 to p_6	10	o_2 to p_5	15	Fetch
1	o_1 to p_2	6	o_2 to p_1	11	o_2 to p_6	16	Transport
2	o_1 to p_3	7	o_2 to p_2	12	o_3 to p_1	17	Screw
3	o_1 to p_4	8	o_2 to p_3	13	o_3 to p_4		
4	o_1 to p_5	9	o_2 to p_4	14	o_3 to p_5		

Table 5.1: Lookup table for the human assembly actions

Operation System (ROS), as illustrated in Fig.5.3a.

The video demonstrations are collected and segmented at intervals of every 45 frames, with the camera operating at a speed of 15 frames per second (FPS). As illustrated in Fig.5.4, the assembly process involves six actions each for o_1 and o_2, and three actions for o_3. Additionally, as detailed in Fig.5.2, two intermediate actions and actions related to screwing are included. Consequently, the action detector considers a total of 18 different actions, as outlined in Table. 5.1.

For the semantic planner, the assembly graph is automatically generated using the data from the trained action detector. This graph is composed of three object nodes, denoted as v_o, and six position nodes as v_p. In this setup, the expert consistently demonstrates the same task structure (i.e., P_o^{pred}) to the robot, regardless of variations in human action sequences. This consistent approach means the robot is trained to always select the first unassembled object from the right, in alignment with the inferred goal P_g^{pred}. The ground truth distribution for P_g^{pred} and P_o^{pred} is provided in the form of one-hot encoded labels. Subsequently, the embeddings of the trained object nodes are annotated, which is essential for training the LSTM.

5.4 Results

5.4.1 Action Detector

The proposed action detector is compared with three baseline methods including:

1. **CNN+LSTM**: The LSTM-FCN is replaced by a simple LSTM to process the

hand motions.

2. **Hand-centric CNN**: The assembly actions are classified with only spatial hand crops via the VGG16 backbone.

3. **CNN-LSTM** [146]: It first processes the image sequences through VGG16 and LSTM is thus used to process the extracted feature sequences. This is a popular framework for video understanding.

Fig.5.5 illustrates the overall effectiveness of the proposed action detector across the entire demonstration video. The detector achieves an overall accuracy rate of approximately 99.5%, showing no particular bias towards any specific class.

Furthermore, Fig.5.6 presents the training and testing outcomes of the described methods, utilizing a training-testing ratio of 0.8:0.2. In these comparisons, the proposed method demonstrates better performance. The integration of LSTM-FCN enhances the detector's ability to handle noisy data. In contrast, relying solely on hand spatial information, as seen in the **Hand-centric CNN** experiment, proves less effective in action classification. Similarly, the **CNN-LSTM** approach also falls short in delivering accurate results. This is due to the presence of extraneous features in the scene. These findings validate that the amalgamation of hand-centric temporal motion and spatial features significantly augments the accuracy in recognizing flexible assembly actions.

5.4.2 Semantic Planner

The previous studies on task planning in HRC, for example, Bayesian Inference [7], Hierarchical task networks (HTN) [66] and AND/OR graph [61], can produce plans as to which object or robot action should be performed. However, these methods are not suitable in the ATO use case. The reasons are:

1. The previous algorithms such as Bayesian Inference, often produce one plan conditioned on prior knowledge. On the other hand, the proposed method is adapt-

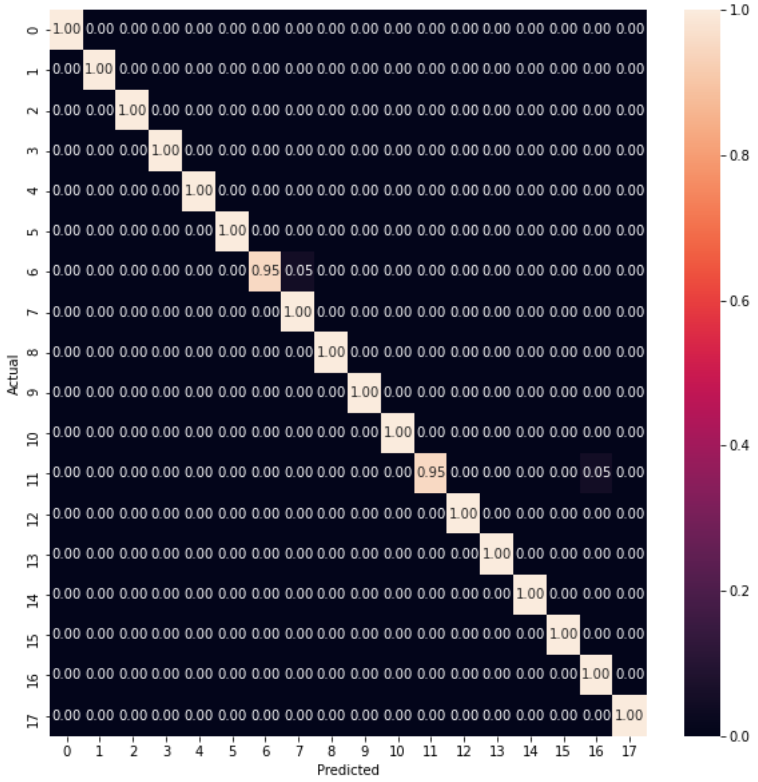


Figure 5.5: Confusion matrix for the proposed action detector on the whole training video data.

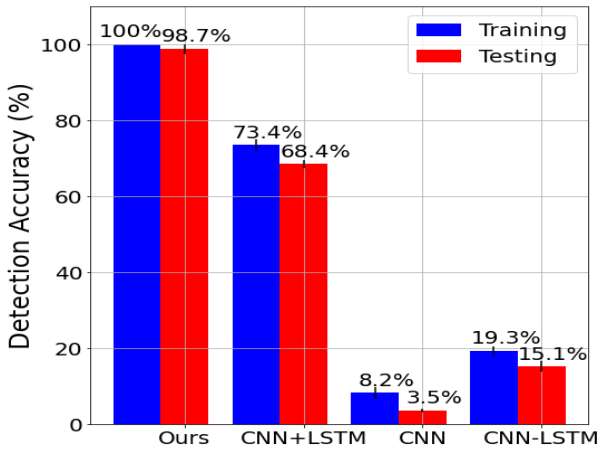


Figure 5.6: Comparison of the proposed action detector with other approaches with average accuracy over five experiments.

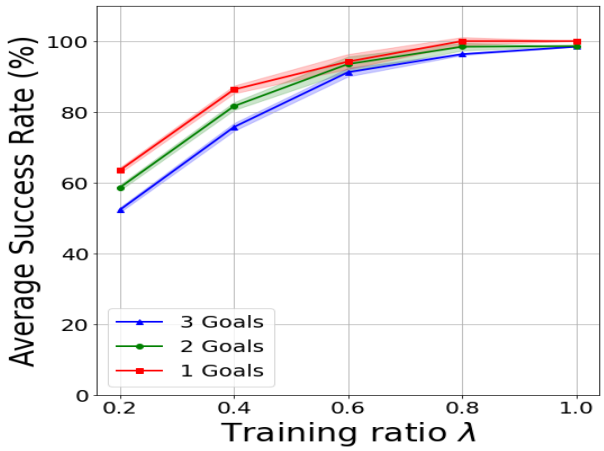


Figure 5.7: The average performance of the Semantic Planner over the training ratio.

able to situations in which actions vary for the robot (i.e. picking one object to multiple positions).

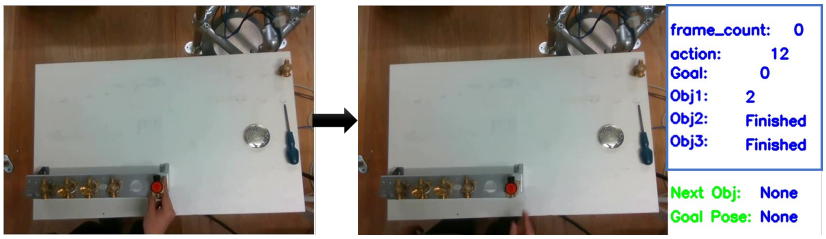
2. There are approaches with symbolic representations, for instance, HTN and AND/OR graph, that can offer feasible plans under one final goal, while the proposed approach can recognise and work for various goals.

3. More importantly, this work is dedicated to releasing the burden of designing the task rules manually in the ATO problem. Through the simulation experiment, it is reported that the graph-based approach is generalizable to unseen human action sequences, which means the planner is capable of producing new plans.

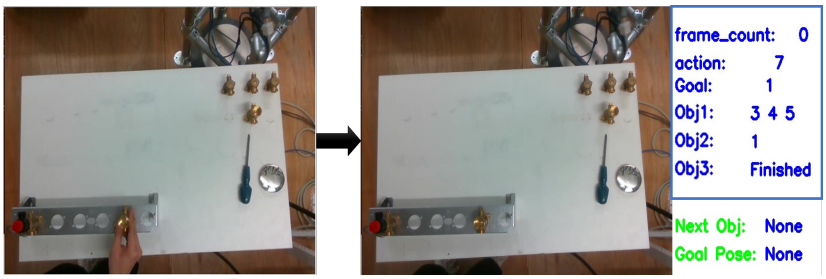
In this assembly framework, the semantic planner can be involved at any stage, leading to ${}_{(m-1)}C_m$ different possible human action sequences for assembling a single final product. For z distinct goal configurations, this results in a total of ${}_{(m-1)}C_m \times z$ potential scenarios. To assess the generalizability of the proposed planner, similar to the previous chapter, the model is trained using only a subset of data randomly selected from all demonstrations. The training ratio λ stands for the percentage of trained data, while the planner's performance is evaluated by the whole dataset. The experiments include the assembly of one, two, or three customized products. To evaluate the model, the Success Rate (SR), which is the ratio of successful trials to the total number of

possible scenarios, is employed. A trial is deemed successful if the predicted $\mathcal{P}_g^{pred}$, $\mathcal{P}_o^{pred}$, and the semantic description txt for each object o are all accurate.

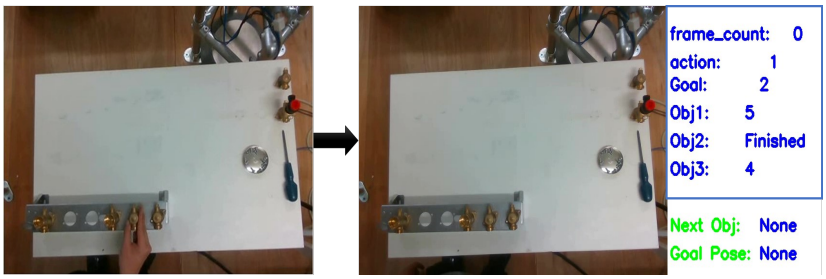
Fig.5.7 demonstrates the GNN-based model's generalizability to scenarios it has not previously encountered. For instance, the model achieves an average SR of 96.3% when trained with 143 demonstrations out of a total of 186 scenarios involving 3 different final goal configurations. The primary instances of failure occur in situations where the same object is assembled to the same position under different goal configurations, such as assembling o_1 to position p_2 for both g_0 and g_1.



(a) Assemble Goal 0



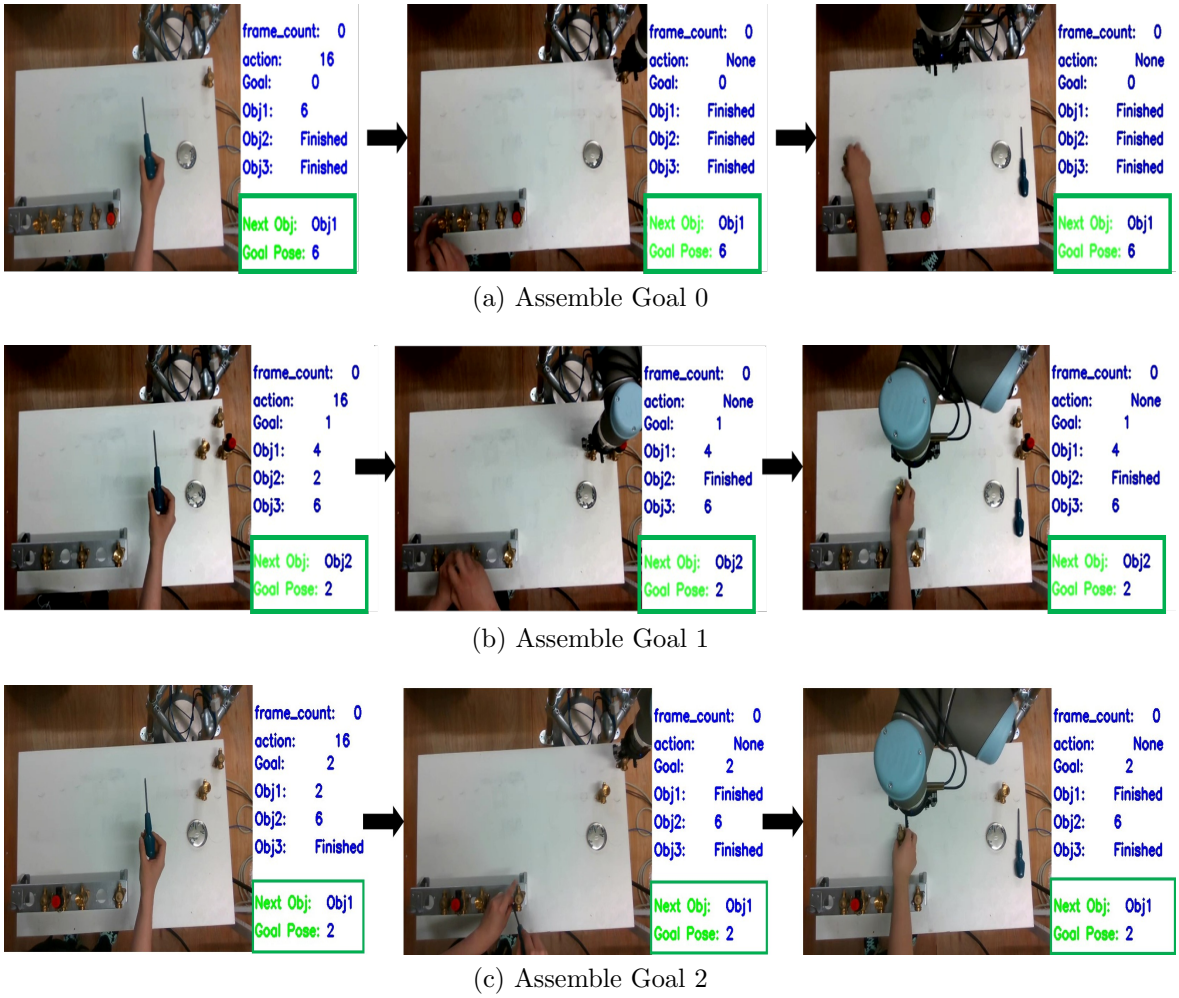
(b) Assemble Goal 1



(c) Assemble Goal 2

Figure 5.8: The results in the real-time experiment. The left figure for each sub-figure is the last detected frame.

5.4.3 Overall performance



(a) Assemble Goal 0

(b) Assemble Goal 1

(c) Assemble Goal 2

Figure 5.9: These are examples of real-time Human-Robot Collaboration (HRC). From left to right, the system first recognizes the human intention of screwing. Therefore, the robot retrieves the next planned object and delivers it to its goal pose (indicated by the green boxes) based on the "Semantic Control" command while the human inserts the objects. The graph and semantic guidance are updated accordingly.

Finally, the whole system's performance is assessed in real-world experiments. The system detects human actions every 45 frames with the camera running over 15 FPS similar to the training phase. The average action detection time is $95ms$ and it will cost an average of $2.1ms$ to plan each object with GPU acceleration.

Moreover, there are 20 experiments for different assembly goals over five times.

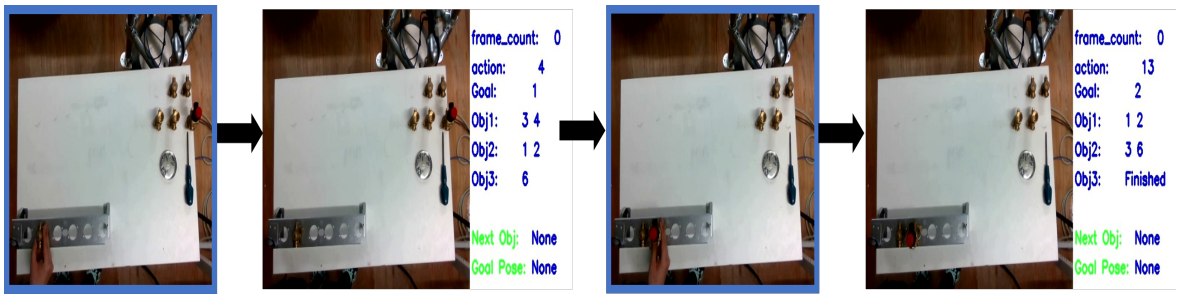


Figure 5.10: This figure demonstrates the proposed system can dynamically construct the graph based on detected human actions. The blue squared pictures are the last detected frame.

The human worker can produce random action sequences. Due to the limitation in the current action detector, fast motions are not well detected. Such errors will cause the decreasing accuracy of the semantic planner.

Fig.5.8 indicates that the proposed system can adaptively guide the human assembly. Moreover, in different goal configurations, a part can occupy the same assembly position. This may lead to confusion for the semantic planner. However, with the further actions of human workers, this error can be eliminated as shown in Fig.5.10. This demonstrates that the proposed system can dynamically correct the wrong predictions by actively updating the assembly graph.

Fig.5.9 also demonstrates the proposed approach can efficiently assist humans by different object handling in real-time. Different shapes of parts o require different grasping strategies in $PickandPlace()$. The semantic output $[o_{next}, p_2, p_3]$ can describe the object types and therefore lead to automated selections of grasping strategies.

Number of goals	AD (%)	SG(%)	SC(%)
1	98.3 $\pm$1.2	97.3 $\pm$1.2	95.3 $\pm$1.2
2	97.7 $\pm$2.1	94.3 $\pm$1.1	93.2 $\pm$1.3
3	95.5 $\pm$1.7	93.8$\pm$1.6	91.8 $\pm$1.6

Table 5.2: The average detection accuracy of Action Detector (AD), the success rate of the **Semantic Guidance** (SG) for human workers and the **Semantic Control** (SC) for the robot produced by the semantic planner.

5.5 Summary

This section concludes the proposed framework and revisits the research questions raised in Section 5.1 based on experimental results.

To address the classification of different human action sequences in **Q1**. The combination of temporal human motions and spatial scene information regarding the object context during assembly can effectively classify various human actions. Moreover, the proposal of using hand-centric information and terming this as hand-object interaction can be efficient, especially for small object assembly. This also can be time-efficient regarding the labelling process when compared to object detection algorithms.

Based on this, the various action sequences can be therefore transferred into an assembly graph. Through the use of graph observation, each object node only holds the relative information of the assembled pose. Thus, the GNN can adaptively infer the human intended goal configurations, which address **Q2**.

Meanwhile, also based on the advantage of the graphic representation, this work further predicts the next object that should be assembled based on the inferred goals and interprets the trained object node embeddings through a simple LSTM. This approach can reveal the relationship between the object and its assembly status, and this relationship can be expressed as human understandable languages. Therefore, the proposed system can work for guiding human future decisions in case of fatigue that may be caused by long-time of work. The semantic relationship also contains different numbers of object positions that should be assembled. This proves that the proposed system can produce more detailed robot plans toward addressing **Q3**. Moreover, the generalization study indicates that the robot has learned the general rules of the demonstrated task structure. Compared to previous studies which heavily rely on prior knowledge from the expert, this work presents a new methodology for HRC that can learn from fewer demonstrations and produce novel plans.

In regards to the limitations of this work, the HRC experiments in Section 5.4.3 were conducted with experts, who are familiar with robotics. Further study with non-expert

co-workers is needed to test out the robustness of the proposed system in real-world applications. This raises another future work of collecting larger-scale and more random human movement data for more efficient human action recognition. Moreover, the current sensing method from only 2D images is unstable when the human moves too fast and hence causes the decreasing accuracy of the overall system. Meanwhile, even though this work can relax the constraints of the pre-defined workflow for human workers, they still need to collect the most reachable objects. This regulation is designed for controlling the robot *PickandPlace* as this work does not use object detections. Moreover, a larger scale of the ATO problem should be studied in the future.

Chapter 6

Iterative Visual Grasping sequence generation for object handling

6.1 Introduction

As noticed in the previous Chapter, the industrial components are well-sorted in the pending area, and the humans still need to follow a regulation to fetch the most reachable parts. This raised the other research theme of how to eliminate such a constraint even when these parts are randomly placed and thus the robot can handle the human requirement flexibly. Take an example from Fig.6.2b, in this cluttered environment, simple object detection and visual servoing approach is inefficient in grasping the object required by humans. As the objects can overlap with each other, their unique geometric shapes can impede the direct grasp of the robot.

Towards addressing this problem, one research direction is to model the geometric property of the target object. Previous works focused on predicting the grasping quality and grasping pose for the end-effector based on the pixels from the object's depth image [106] [107], point cloud data[109], or event-based camera [110]. Wang et al. [111] also concerns the generation of the collision-free trajectory generation. They proposed a hierarchical framework that learns goal-driven grasps based on partial point

cloud observations. Moreover, some other research focuses on using haptic information rather than visual features. Chen et al.[147] adopts tactile force information with deep-learning networks to estimate the number of objects in hands. Abi-Farraj et al.[148] describes a novel tactile shared control method to assist human operators in sorting and segregating multiple targets in cluttered and unknown environments. The aforementioned methods often require rich information about the target object, It can be problematic when the target object is overlapped by the surrounding objects too much and therefore the direct grasping pose estimation may not be effective.

Another possible solution is to reason the objects' relationship through the visual feature, and thus allow the robot to grasp the intended object orderly. Current research focuses on classifying the visual relationship between each object pair in a spatial image[112],[113],[114],[115],[119],[120] or from a temporal video clip [116]. For the specific application in robotics, Goodwin et al.[117] investigated the rearrangement of unseen but similar objects with seen goal images by leveraging the semantic and visual information through CLIP. Moreover, graphic observations, which can naturally hold the spatial object relationships, have been investigated in [118] [110], [121],[122]

Similar to the object stacking environment studied in [119],[120],[121],[122], this chapter studies the object handling from the cluttered environment through the 2D image. From the above discussion, this chapter proposes three research questions that should be addressed for the specific object-handling problem:

- **Q1**: Unlike the previous works in 6D object grasping, this work considers the limited capability of the end-effector. For the occluded target object, there should be a feasible plan to make such an object become graspable instead of directly grasping.

- **Q2**: The aforementioned works often focus on recognising the relationship between each object pair. As for robotics manipulation, there should be extra effort in building the manipulation tree through logic operations. On the contrary, this work aims to directly produce the manipulation rules from the encoded graphic

scene.

- **Q3**: Moreover, considering the realistic scenario when the human only requires specific types of object. This work aims to develop a framework that can be generalizable when the request switches from time to time. This means that the framework is only trained with specific goals, while during testing, it should produce a new plan for the unseen goals.

Towards addressing these questions, the chapter proposes a visual grasping system that can iteratively generate the solution to manipulate the object orderly. The porposed framework can be described as follows:

- Based on the specific object requirement made by humans, the proposed framework first encodes the raw visual image to graph observation through a pretrained detection network. Through the graph construction, unlike the previous work in [121],[122], where they use a pre-defined distance threshold in image space to filter out the irrelevant object pairs. This work directly aggregates the objects' features within the image through a weighted edge. This is to prevent the lack of information in this specific scenario. Thus, the GNN is able to process the visual feature and evaluate the spatial relationship between the required objects and their neighbours. As a result, the graspability of the required object can be classified.

- If there are no graspable objects, this requires the robot to generate feasible solutions such as moving the surrounding objects or obstacles by decoding the trained object node embeddings into contextual languages.

The proposed approach has been validated in real-world experiments and daily life object datasets. The results have shown that the proposed approach is capable of directly producing a manipulation solution by recognizing and prioritizing the robot's actions based on the objects' geometric properties. This approach can be also generalizable for unseen images and requirements.

6.2 Methodology

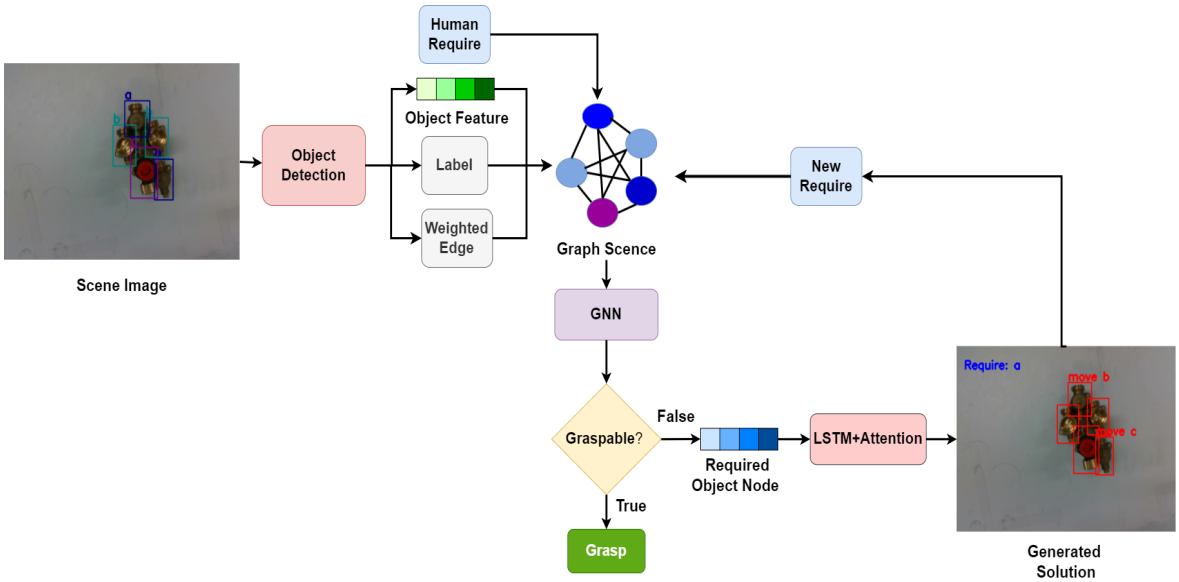


Figure 6.1: Pipeline of the proposed framework: The raw scene image will be first processed through a common object detection network (e.g. Faster-RCNN). The detected objects' visual features will be used to construct a graph observation. According to the human demand, the GNN should first produce the graspability of the inquired object while considering the robot's capability. If it is not graspable, an LSTM with Attention will propose the solution as " move which objects" to enable the demanded object to be graspable. It will be thus used as the new demand. This process will continue until the demanded objects are graspable.

Given an image scene containing various objects, the goal of this study is to enable the robot to identify the graspability of the object required by humans. Considering the cluttered environment as shown in Fig.6.1, the graspability of an object is determined by both its surrounding objects and its own geometrical properties, while respecting the capability of the robot. For example, considering a robot can only grasp a required object from the top, if such an object is overlapped by a taller object, it is considered as not graspable at the current stage. Therefore, this work aims to provide a solution to remove the obstacles iteratively until the required object can be grasped.

Fig.6.1 describes the proposed framework. It first uses a Faster R-CNN [149] with

Resnet101 as the backbone feature extractor for object detection. The obtained objects' features proposed by Faster R-CNN including its feature values $\mathbf{f}_i$, label l_i, and the 2D center position of the bounding box $\mathbf{d}_i$ within the 2D image. Relying solely on spatial information of objects within an image, such as bounding box coordinates, does not adequately convey the graspability of the demanded object in the given scenario, for example, a deterministic distance threshold.

Instead, this study encodes the detected objects into a graph observation. Suppose graph observation $\mathcal{G}$ containing n nodes as $V = \{v_0, .., v_1, v_2, ..., v_n\}$. Each node contains the extracted features $\mathbf{f}_i$, object label l_i and the binary goal feature indicating the demand from human g_i, $v_i = \{\mathbf{f}_i, l_i, g_i\}$

The graph is fully linked with directed edges. Unlike the previous works in this thesis, which consider the temporal changes in the environment, such as positions, and human actions. The spatial relationship of the objects is reflected in the edges. Consequently, in the construction of the graph observation, the normalized inverse weighted edge $E = \{e_{ij}\}$ is employed to articulate the relationship between each object and its adjacent objects as shown in Eq.6.1, where $\|\mathbf{d}_i - \mathbf{d}_j\|$ is the 2D Euclidean distance between the object node v_i and its neighbour node v_j based on the centre point of the bounding box.

$$e_{ij} = \frac{\frac{1}{\|\mathbf{d}_i - \mathbf{d}_j\|}}{\sum_{j=1}^{n} \frac{1}{\|\mathbf{d}_i - \mathbf{d}_j\|}} \tag{6.1}$$

To process such a graph observation, the Weisfeiler-Lehman (WL) inspired graph neural network operator (WL-GNN) is adopted [150]. The WL algorithm is common method for Graph isomorphism testing, which determines whether two graphs have the same structure. It iteratively updates node labels through multiple rounds of neighbour aggregation. The WL algorithm takes into account the multi-hop neighbourhood information of nodes, enabling it to capture the intricate structures within a graph.

Inspired by this, WL-GNN incorporates information from both the node itself and its multi-hop neighbours:

$$\Gamma_i^{k+1} = \sigma\left(W_1^k \cdot \Gamma_i^k + W_2^k \cdot \sum_{i \in \mathcal{N}(i)} \cdot e_{ij} \cdot \Gamma_j^k\right) \tag{6.2}$$

Γ_i^k is the target node embedding at k_{th} message passing and Γ_j^k is the neighbour nodes embedding. Moreover, $W_1^{(k)}$ and $W_2^{(k)}$ are the weighted matrix to handle the information from the current node and its neighbours, which ensures that during the information aggregation process, the information from the current node and its neighbours is kept distinct. During message passing, the weighted matrix e_{ij} is used to modulate the propagation of information between nodes. The neighbour node with a higher weight edge can have a greater influence on the target node. Unlike Graph Convolution Networks (GCN), where node representations are based on their immediate neighbours WL-GNN is more capable of exploring richer and more in-depth graph structural information.

Afterwards, the graspability is considered as a node classification problem with binary output as P_{pred}^{gra}. The Cross-entropy loss is used to optimize the model:

$$loss = -[P_{goal}^{gra} \log\big(P_{pred}^{gra}\big) + (1 - P_{goal}^{gra}) \log\big(1 - P_{pred}^{gra}\big)] \tag{6.3}$$

For objects identified as non-graspable, similar to prior work, the trained nodes not only retain information about themselves but also about their weighted neighbours. Additionally, during the learning process within the graph, it appears that the graph captures the demonstrated task structure. Consequently, the object node embedding is decoded into its corresponding solution as contextual information through Long-Short-Term-Memory with Attention Mechanism (LSTM_Att) [151].

The input feature consists of the trained node embedding Γ_i^k, object label l and the graspablity P_{pred}^{gra} as $f^{'} = \{\Gamma_i^k, l, P_{pred}^{gra}\}$. Given the current LSTM hidden state h_t and the input feature, the attention mechanism first computes an attention score as Eq.6.4, where W_k is the trainable weights and $f_s^{'}$ is a segment of the input feature. Accordingly, the attention weight can be expressed as Eq.6.5. As a result, it provides

a context-based weight to different portions of the input feature as shown in Eq.6.6.

$$\text{score}(h_t, f^{'}) = h_t^T W_a f_s^{'} \tag{6.4}$$

$$\alpha_{ts} = \text{softmax}(\text{score}(h_t, f_s^{'})) \tag{6.5}$$

$$context_t = \sum_s \alpha_{ts} h_s^{'} \tag{6.6}$$

Therefore, $context_t$ is the new input feature of LSTM at time step t. A similar training process can be applied as described in the previous section.

The original intent behind the Attention mechanism was to assist models in determining which part of the encoded sequence to focus on during decoding. In this task, this implies identifying which words of the input sentence should be emphasized when generating a particular word in the output. Therefore, the model needs to understand the semantic relationships between words, and word embeddings provide this semantic information to the model during testing as shown in Eq.6.7. The initial word embedding **embeds** will always be the "start" token.

$$txt_i = LSTM\_Att(\Gamma_i^k, l, P_{pred}^{gra}, \textbf{embeds}) \tag{6.7}$$

Algorithm 4 describes the proposed iterative visual grasping framework. The initial requirement is made by humans. Consider the complex scenario when such an object is stuck in multiple objects. Thus, based on the extracted visual feature, the proposed framework aims to iteratively generate solutions on how to remove these obstacle objects by using it as the new requirement or input of the G and the $LSTM\_Att$, if such obstacle objects are still classified as not graspable. For a single image or scene, it aims to find the most graspable object o_{re}. For robot grasping, the ultimate goal is to obtain the human-required object o_{goal}. Therefore, this process may involve different

steps of solution generation.

Algorithm 4 Proposed Iterative Grasping System

Initialize the object detector O

Trained GNN model G

$LSTM\_Att$ decoder

Input: human required object o_{goal}

while *not* P^{gra}_{goal} **do**

  Obtain the raw 2D image I

  Extract visual information $f, l, d = O(I)$

  Construct the graph scene $\mathcal{G}$ according to o_{goal}, f, l, d

  Predict the graspability of the required object $P^{gra}_{goal} = G(\Gamma^{k}_{o_goal})$

  if *not* P^{gra}_{goal} **then**

    Find the new require object $o_{re}, ..., = LSTM\_Att(\Gamma^{k}_{o_goal}, l, P^{gra}_{goal})$

    while *not* P^{gra}_{re} **do**

      Construct the graph scene $\mathcal{G}$ according to o_{re}, f, l, d

      Predict the graspability of the required object $P^{gra}_{re} = G(\Gamma^{k}_{o_re})$

      if *not* P^{gra}_{re} **then**

        Find the new require object $o_{re}, ..., = LSTM\_Att(\Gamma^{k}_{o_goal}, l, P^{gra}_{re})$

      else

        Robot Grasp

        break

6.3 Experimental Setup

To validate the proposed system, there are two experiment environments have been set.

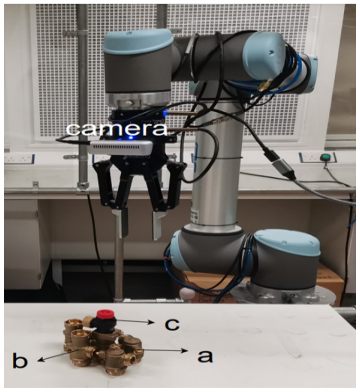
The first experiment considers the handling of industrial components within a cluttered scene. In this study, objects can be randomly oriented, and multiple components might be overlapped with each other. A Universal Robot 10 (UR10) arm, equipped

with a camera at its end-effector, should primarily capture 2D images from a top-down perspective and execute a top-to-bottom grasping action based on human requirements as shown in Fig.6.2a. In this experiment, there are three types of industrial components including: "a,b and c". The maximum amounts for each type in the scene can be 3,2 and 1 respectively as shown in Fig.6.2b.

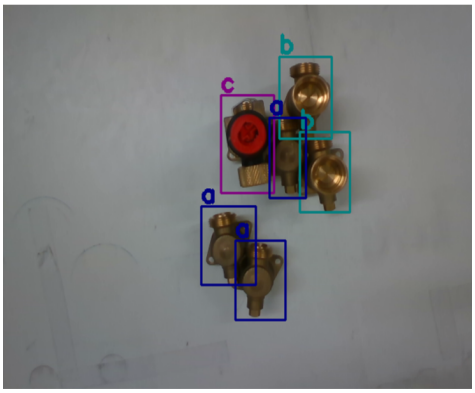
A two-finger gripper robitq2F-85 is utilized for robot grasping. Only 3D position information respected to the robot base is provided via a depth camera RealSense D435i. This means the orientations of the objects are not considered, and the gripper is constrained during the grasping as shown in Fig.6.2c. Throughout this study, consideration is given to the geometric shapes and heights of each component. The task requires the robot to determine the most appropriate object that can be grasped. If the required objects are classified as ungraspable at a given scene, there emerges a need to evaluate their neighbour parts and the spatial relationship they share. And therefore generate robotic solutions as to how to make the required object become graspable. For this, the main objective is to allow the robot's ability to recognize and prioritize its grasping actions based on the geometric and spatial characteristics of the objects presented. For example, object c is the most graspable one as it is taller than the other two in this study. Moreover, considering the different orientations of the components, the deterministic distance threshold (e.g. 2D distance between the bounding box) is inefficient as shown in Fig.6.3. Therefore, capturing and extracting the visual information of the objects is necessary.

In regard to the training process, the objects' visual information is first labelled and extracted by Faster-RCNN with Resnet101 as the backbone with feature pyramid networks (Faster-RCNN-fpn) which is written in pytorch Detectron2 [152]. Based on this information, the graph can be constructed with human requirements accordingly. The binary ground truth labels for training GNN are provided by human expert where 1 stands for graspable and 0 stands for ungraspable. For the required objects classified as ungraspable, their trained node embeddings are annotated with solutions. The solutions include: "Move b and/or c, help".

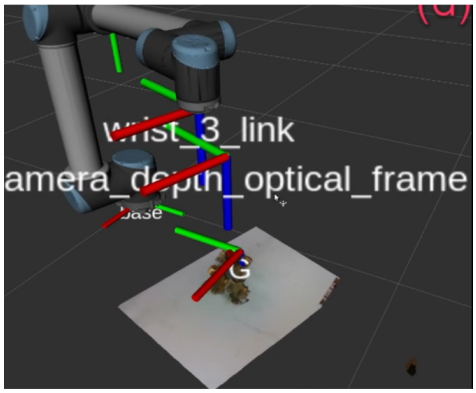
All the models being trained and tested in the following experiments are written in Pytorch and PyG with a GPU acceleration. The robot is controlled by the Robotics Operation System (ROS) and MoveIt.



(a) Physical setup



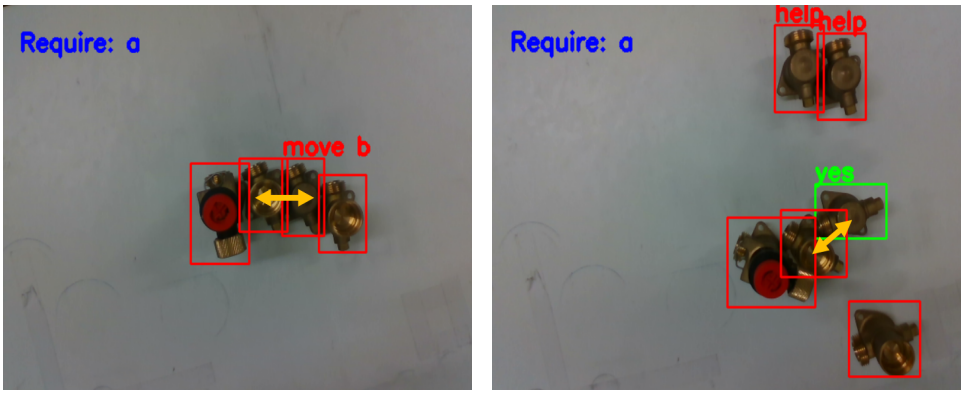
(b) Robot view of the cluttered scene



(c) Robot view of the cluttered scene

Figure 6.2: The industrial components handling setup

Moreover, the proposed architecture has also been validated in the open-source dataset Visual Manipulation Relationship (VMR) V2 Dataset [122]. This dataset contains an object's stacking environment as shown in Fig.6.4. Similar to the first experiment, the main task is to produce the manipulation plan for the intended goal object.



(a) (b)

Figure 6.3: The double-headed arrows show the distance between the objects. The distance between part a and part b (0.08) in Fig.6.3b is smaller than the distance (0.10) in Fig.6.3a. The green box stands for graspable objects while the red boxes stand for ungraspable objects.

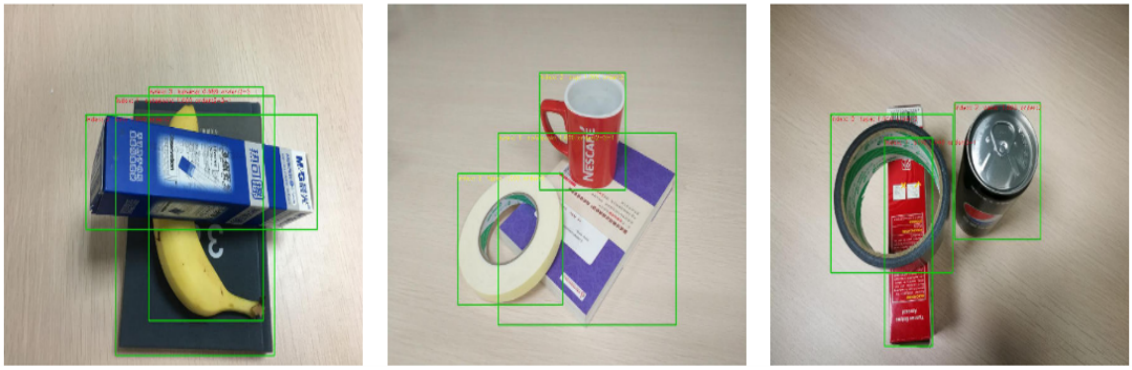


Figure 6.4: Examples from VMR dataset for stacking environment.

6.4 Results

6.4.1 Industrial parts handling in cluttered environment

This experiment is designed to demonstrate the generalizability of the proposed framework. Considering that human requirements can vary from one to another, for each training image, there will be randomly selected object goals or requirements, meaning that not all the object types shown in the image scene will be labelled. However, the trained framework will face various unseen scenarios including unseen goals or totally

unseen images during testing.

There are 427 training samples with randomly selected goal objects and 50 testing images with various types and numbers of objects. The proposed system is compared with several different approaches. The compared approaches are listed as follows:

- **GNN_Att w/o WE**: This is an ablation study. The WL-GNN processes the nodes' features without the proposed weighted edges (i.e. all weights equal to 1).
- **GNN_LSTM**: The LSTM_Att is replaced by a simple LSTM with the same parameter settings.
- **GNN_DF_Att**: The Distance Filter (DF) aims to filter the irrelevant object pairs according to a pre-defined imaged-based Euclidian distance threshold, which is set to 0.16 in this work. This is to demonstrate the importance of processing the visual features when the orientation of objects varies.
- **GVMRN_RF** [121] [122]: The previous work adopts Graph Convolutional Neural Network (GCN) to perform the relationship reasoning between each object pair. Unlike the proposed framework, it extracts the union box features, which can cover two overlapped objects' bounding boxes, as node features. Thus, the relationship reasoning can be termed as a multilabel classification problem with labels as "under, above, help, no relation". To filter out the irrelevant objects, they further proposed a Relation Filter (RF) based on the intersection area and a pre-defined distance threshold. Moreover, in order to ensure consistency in the testing criteria, only the relationship nodes relevant to the required objects have been trained. As it is not able to directly produce a grasping solution as contextual language, the prediction accuracy is assessed.
- **GCN_Att**: To further compare with **GVMRN_RF**, the WL-GNN is replaced by GCN in the proposed framework.

The object-based accuracy (OA) is first assessed as if the predicted graspability and generated solutions for ungraspable objects are correct based on one single image.

There are three scenarios being assessed as randomly requiring 1,2 or three types of objects. Considering the geometric shapes, the grasping solutions can vary. For example, the requirement of the object "a" can be the most difficult case, where its accuracy is around 0.81. This is because it is the smallest part and may therefore the generated solution can include both "b" and "c" objects as shown in Fig.6.6.

There is also a special case when two same type of objects get too close, the generated solution will ask for "help" from human co-worker as the robot is not capable of grasping it by removing other objects. Fig.6.6a and Fig.6.6b illustrates successful testing cases to find the most graspable object based on human requirements.

Table 6.1 describes the OA in a single image for different approaches. It indicates the weighted edges can improve the aggregation process of the GNN and therefore provide more effective features for decoding in the study of **GNN_Att w/o WE**. Meanwhile, the LSTM_Att shows a slight improvement when compared to simple LSTM. For the results in GNN_DF_Att, it is found that a pre-defined distance threshold can not fully describe the spatial relationship between each object. Fig.6.6 describes a failure testing case with DF, where the distance between object "a" and object "c" is greater than the pre-defined DF. This will cause incomplete solution generation.

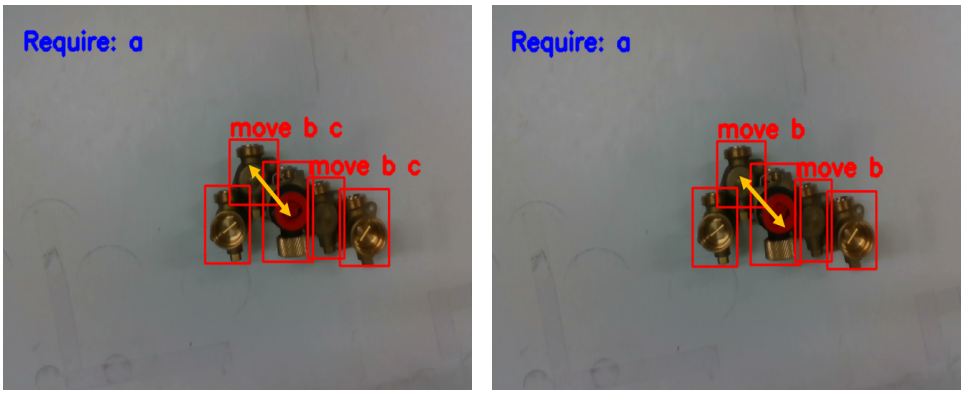
In the comparison of **GVMRN_RF** and **GCN_Att**, the first finding is that their architecture has worse performance in terms of generalizability. The main reason is that the relationship node feature is totally new during testing, and thus it can not efficiently predict labels. While the proposed architecture only modifies parts of the node features (i.e. goal information) when a new requirement is set. Meanwhile, it was also found that GCN could not extract informative spatial features as shown in the results of **GNN_Att** and **GCN_Att**. The reason is that GCN performs only average aggregation over the node embeddings and its neighbours.

To further assess the proposed system's ability to iteratively generate grasping solutions, two more experiments have been conducted as shown in Table 6.2.

As mentioned above, for one initial image, if none of the required objects is graspable, the proposed system aims to iteratively propose the most graspable object as

the solution, which can be illustrated in 6.7a and Fig.6.7b. Therefore, the first experiments are conducted to manually assess if the grasping solutions are corrected for the goal objects from one initial image. There were 10 testing cases for different numbers of objects. As shown in Table 6.2, the accuracy drops down along with the increase in the number of objects.

Secondly, the proposed system is integrated with a real robot arm for the real-world grasping task. There were also 10 testing cases for each number of objects. As shown in Fig.6.8, the proposed system has the capability of dealing with variant numbers, and object types and there will be different task lengths depending on the initial scene. Meanwhile, it also has the ability to generate new solutions for new unseen images, which can occur after robot grasping. However, it has been noticed that the gripper may accidentally collide with the surrounding objects and may lead to a decrease in performance. This reveals one drawback of the proposed system: it is not capable of recognising visual features with too-large orientation variations in the object and these variations can lead to infinite possible scenarios.



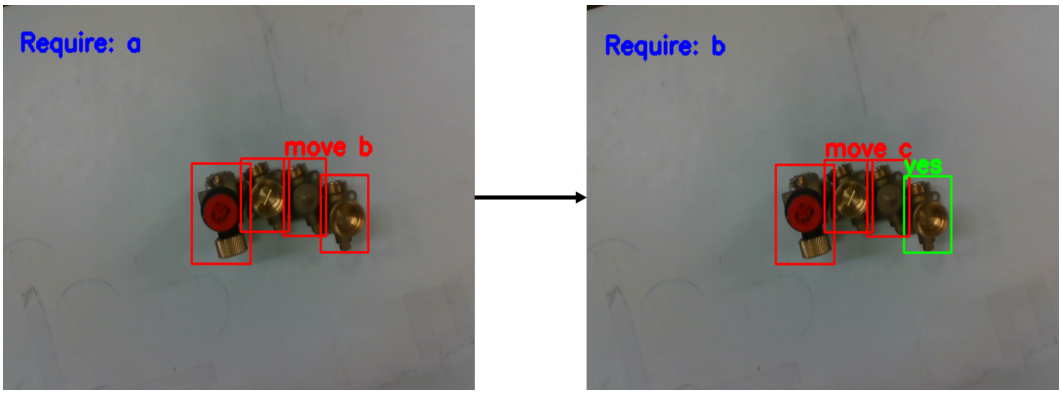
(a) Grasping solution generated by the proposed GNN_Att

(b) Grasping solution generated by the **GNN_DF_Att**

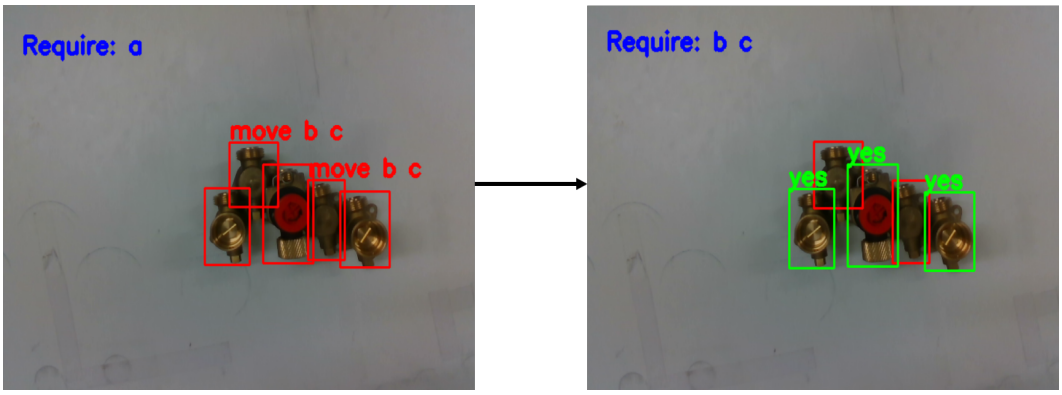
Figure 6.5: Comparison between the proposed method and GNN_DF_Att.

(a) Require object a (b) Require object a,b

Figure 6.6: The proposed architecture performance on one single image.



(a) Require object a



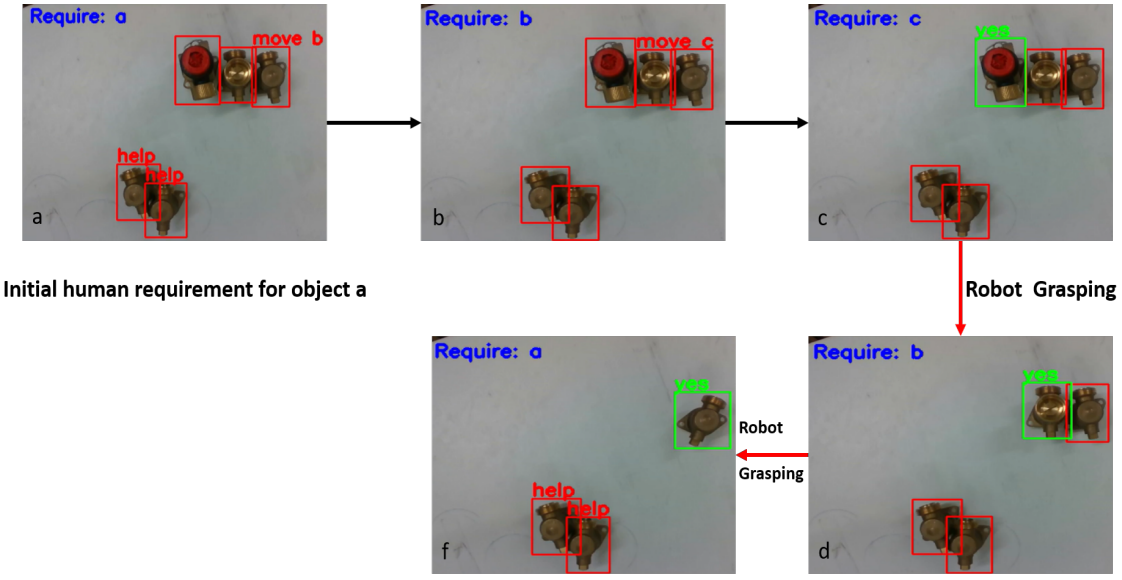
(b) Require object a

Figure 6.7: Iterative grasping solution generation from the initial image.

	Train_Acc	One type	Two type	Three type
GNN_Att	1	0.91 ±0.018	0.891 ±0.021	0.882±0.016
GNN_Att w/o WE	0.97	0.634 ±0.013	0.586 ±0.004	0.545 ±0.003
GNN_LSTM	1	0.89 ±0.007	0.872±0.017	0.864±0.014
GNN_DF_Att	1	0.881 ±0.015	0.843 ±0.008	0.829 ±0.005
GVMRN_RF	-	0.393±0.023	0.467±0.019	0.544±0.018
GCN_Att	0.738	0.421±0.025	0.621 ±0.022	0.692 ±0.022

Table 6.1: Object Accuracy (OA) comparison between different approaches in single image. Train_Acc stands for the graspability classification in training sets. Meanwhile, the performance based on different numbers of human requirements has been shown.GNN_Att stands for the proposed algorithm.

(a) Require object a



(b) Require object a

Figure 6.8: Physical robot experiment scenes for dealing variations including object numbers, object types, and task lengths. As the figure shows, the framework allows the robot to remove the graspable obstacle objects until it identifies the graspable object required by human.

	3 objects	4 objects	5 objects	6 objetcs
Single image	0.962	0.901	0.864	0.806
robot grasping	0.91	0.874	0.782	0.684

Table 6.2: The accuracy for iterative visual grasping despite the object detection error

6.4.2 Daily life objects in stacking environment

In this experiment, two baseline methods have been compared in the VMR dataset as follows:

- VMRN [153]: This is a CNN-based visual manipulation reasoning network. The objects' features are first extracted by the backbone object detector. Therefore, they need to classify the relationship between every possible object pairs through CNN.

- GVMRN_RF [121] [122]: The previous work adopts Graph Convolutional Neural Network (GCN) to perform the relationship reasoning between each object pair. The relationship reasoning can be termed as a multilabel classification problem with labels as "under, above, help, no relation" through the union box features. To filter out the irrelevant objects, they further proposed a Relation Filter (RF) based on the intersection area and a pre-defined distance threshold.

Image-based accuracy is assessed in this experiment. IA stands for if the objects' relationships are all correct within one image. However, unlike the proposed framework, these two works only classify the relationship between different parts. As for the robot manipulation, they need to traverse the neighbour objects related to the target objects. Thus, to compare, IA in the proposed framework is assessed if the graspable object is classified correctly and solutions for the ungraspable object are correct.

Table 6.3 describes the IA performance in the testing dataset. The testing dataset contains 31 types of different daily life objects and the maximum number of objects in one image is 5. As the table shows, the proposed framework performs worse than the baseline algorithms.

To figure out the reason behind this issue, further study is carried out. It has been found that the main failure case happens as the generated solutions often produce the wrong name or label of the objects. Fig.6.9 illustrates the IA performance on both GNN classification and solution generated from LSTM_Att over the increasing number of types of objects. It has been found that GNN performance does not decrease a lot while the LSTM_Att's performance drops. Therefore, it is suspected that the main reason for poor performance is that LSTM_Att can not decode the features efficiently as the variety of objects grows. Moreover, considering less orientation can happen in this dataset, the RF method in GVMRN_RF can be more effective in aggregating relevant object information.

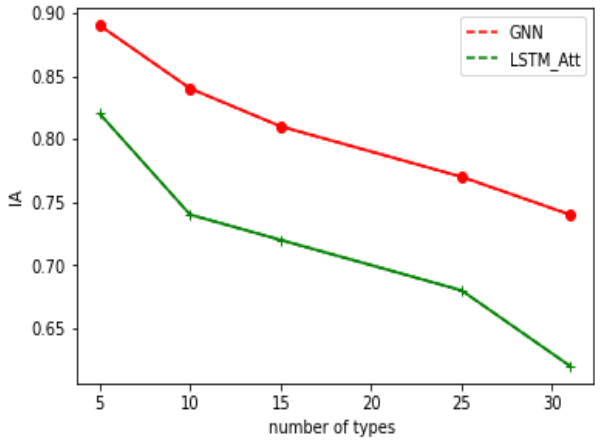


Figure 6.9: The average performance of GNN and LSTM_Att over the increasing number of types of objects.

	VMRN	GVMRN_RF	GNN_Att
IA	0.658	0.688	0.62

Table 6.3: Image-based Accuracy for different methods in VMR dataset.

6.5 Summary

This section concludes the proposed framework and revisits the research questions raised in Section 6.1 based on experimental results.

This study assumes that the robot is constrained to only execute top-down manipulation for the objects. Through the experimental study, it has been found that due to the orientation variation that may exist in every object, visual features play an important role in graspability detection. By leveraging the visual features and the symbolic object information (e.g. object label and goal property) through the GNN, the proposed framework can efficiently detect the graspability of the target object while considering the spatial relationship it may share with neighbour objects within the scene. Moreover, the weighted edges also play an important role when aggerating the informative neighbour features. Afterwards, the proposed embedding-to-sequence approach using an LSTM-like neural network allows the robot to generate actions or plans by recognizing and prioritizing the geometric and spatial characteristics of the objects. This can address **Q1** and **Q2**.

To answer **Q3**, the proposed graph structure tends to be more generalizable than the previous works. The new goals only modify the symbolic node feature rather than directly using the total unseen relationship nodes, for example, unseen union box features.

Regarding the limitation of the work. In the industrial object handling scenarios, it can be found that the current approach is still unable to handle the too-large orientations of the objects in both GNN and LSTM_Att during the test. Moreover, in the daily life object stacking environment, it shows worse performance than the previous works. The main reason can be that the current decoding model such as LSTM_Att is still inefficient in processing the node features.

Chapter 7

Conclusion and future work

7.1 Conclusion

This thesis has focused on developing novel frameworks that are dedicated to robot control strategy to enable adaptive HRC in personalised MTO manufacturing scenarios. It studied different types of HRC, including sequential HRC and responsive HRC, with different information sources including 3D positions information and visual features. All the proposed frameworks aim to offer generalized robot abilities that can adapt to variations from either human performance or environmental variations through intuitive training methods.

Chapter 3 first studied continuous trajectory learning in a sequential HRC setting. Compared to the current probabilistic approaches in LfD, the main advantage is that the proposed motion planner separates the motion generation into two steps as subgoal planning and action generation. By accessing the task stage information, the proposed method is more capable of integrating diverse demonstrations for different initial and goal positions. The idea of modelling the dynamic of expert preference in terms of goal-state distance can effectively produce joint action without extra exploration, This shows the generalization ability of the proposed approach regarding position variations. However, the proposed motion planner still suffers from high-dimensional data. A

hybrid guidance framework that combines LfD and DRL is proposed. The results show that the exploration issue of DRL can be addressed by providing guidance from LfD. And therefore improve the task success rates. While it should be noted that, there should be a trade-off between exploration and exploration in order to maintain the experience diversity for DRL.

From Chapter 3, the main issue in multi-object tasks is high-dimensional observation for probabilistic-based subgoal planning. And the DRL agent still needs long time training. Therefore, inspired by natural human thinking, Chapter 4 further proposed an end-to-end TAMP framework. The Graph-based reasoning module learns the demonstrated task structure by ranking the most important observations under different task stages including picking and placing. It is adaptive to the changes in task structures, which can be caused by human performance. Moreover, the study in practical scenarios also shows its advantage in terms of object changes. Compared to previous studies, it can learn the underlying task structures without any hand-coded manipulation rules and can be more generalizable to task structures in terms of sequential HRC. It can thus effectively reduce the observation dimension for the motion planner. The results of the motion planner also illustrate its capability of generalizing collision-free trajectories in new tasks. The overall TAMP system trained in simulations can be directly applied to the physical world as long as the distribution of the input stays similar.

Chapter 5 proposed a vision-based planning system for flexible responsive HRC. Unlike the works in Chapter 3 and 4, where human performance and goal configurations are reflected by the objects' positions, this work infers them from visual human information. In order to achieve this, a hand-centric action detector is proposed. Instead of using object detection with a heavy labelling process, this detector combines the spatial human assembly context as hand-object interaction and temporal hand motions to better describe the human actions as which objects have been assembled to where. Furthermore, these detected human action sequences can be dynamically transformed into graphic observations as the relationship between objects and assembly positions.

The proposed semantic planner can thus infer the human intended goal configuration and the planned object which should be handled by the robot. Furthermore, it can provide a more detailed robot action as to which object should be handled to where by the robot. Compared to previous works, this work can work under various goal configurations with various human action sequences. More importantly, the previous works often heavily rely on the prior knowledge of the task structure, while this work is more capable of generating new plans when human action sequences are novel.

In Chapter 5, as there is no visual information on the objects, robots should still follow regulations on handling the objects required by humans, so as to human coworkers. Therefore, Chapter 6 aims to address this constraint in an even more complex environment. This work assumes the assembly object or parts are cluttered while the robot's grasping capability is constrained. To address this problem, this chapter proposed an iterative visual grasping solution generation framework. Firstly, the detected objects visual features are encoded into spatial graphic observations. The graspability of the objects demanded by humans can be classified. For ungraspable objects, it can iteratively generate solutions as separate obstacle objects via decoding the trained objects' node embeddings. The results showed that the proposed system is able to produce solutions or robot actions while respecting the geometrical characteristics of objects and the weighted spatial relationship they may share with surrounding objects. Compared to the previous works, the proposed system is able to directly produce solutions or robot actions even with unseen human requirements and unseen image scenes.

7.2 Future work

This section discusses the potential future works based on the experimental results and limitations identified previously in this thesis.

In Chapter 3 and 4, the proposed TAMP system only considers object manipulations in 3D space, which can be problematic when the objects need a proper grasping solution, for example, deformable objects. Thus, the potential work can utilize depth

information such as orientations. Moreover, the current TAMP system only considers static obstacles. However, in the real world, dynamic obstacles could happen. This is also related to safe collaboration, for example, humans can move dynamically in the workspace. The potential future work could consider the detection of human movement and actively infer the optimal joint actions. Moreover, instead of representing human performance as object positions in the current TAMP system, it could also 'image' the possible consequences which can happen to the current task by using human motions. These further directions could facilitate a better human-robot-cooperation simultaneously.

In Chapter 5, the main issue of the vision-based semantic planner is in the human action understanding approach, which is not robust enough for fast human motions. Therefore, one possible work is to utilize wearable sensors to minor the motions and may adopt tactile pressure sensor gloves to detect hand-object interactions. This could require multimodal sensor fusion technologies. Moreover, more comprehensive studies on the larger scale of the ATO problem should be considered to test the proposed framework's robustness.

In Chapter 6, the current visual grasping solution generation framework can not outperform the state-of-the-art approaches in the stacking environment. The main reason could be the embedded features are not well-represented and the LSTM-like models can not generate the solutions when object types are too much. One potential work could utilize depth images or point cloud information instead of 2D images. Moreover, the overall accuracy can also be improved using Transformer-based decoding models.

Moreover, the proposed frameworks in this thesis are all works in the multi-objects under the same task family. To further develop a generalized robot platform that can work across different HRC scenarios in different tasks, current advances in larger visual-language models could be one direction [154], [155]. One future work is to understand human actions as languages while considering temporal image features, and thus produce robot actions.

Bibliography

[1] E. C. Townsend, E. A. Mielke, D. Wingate, and M. D. Killpack, "Estimating Human Intent for Physical Human-Robot Co-Manipulation," *arvix*, vol. abs/1705.1, 2017.

[2] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, no. February, pp. 248–266, 2018.

[3] International and F. of Robotics, "Demystifying Collaborative Industrial Robots," *International Federation of Robotcis*, no. October, pp. 2 – 3, 2019.

[4] S. Benjaafar and M. Elhafsi, "Production and inventory control of a single product assemble-to-order system with multiple customer classes," *Management Science*, vol. 52, no. 12, pp. 1896–1912, 2006.

[5] E. C. Morley and C. S. Syan, "Teach pendants: How are they for you?," *Industrial Robot*, vol. 22, no. 4, pp. 18–22, 1995.

[6] P. Neto and N. Mendes, "Direct off-line robot programming via a common CAD package," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 896–910, 2013.

[7] Y. Cheng, L. Sun, C. Liu, and M. Tomizuka, "Towards Efficient Human-Robot Collaboration with Robust Plan Recognition and Trajectory Prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2602–2609, 2020.

[8] Maria Fox and Derek Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, vol. 20, pp. 1–48, 2003.

[9] Z. Wang, R. Qin, J. Yan, and C. Guo, "Vision sensor based action recognition for improving efficiency and quality under the environment of industry 4.0," *Procedia CIRP*, vol. 80, pp. 711–716, 2019.

[10] H. Nguyen and H. La, "Review of Deep Reinforcement Learning for Robot Manipulation," *Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019*, pp. 590–595, 2019.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[12] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 819–825, 2017.

[13] J. Xu, Z. Hou, W. Wang, B. Xu, K. Zhang, and K. Chen, "Feedback Deep Deterministic Policy Gradient with Fuzzy Reward for Robotic Multiple Peg-in-Hole Assembly Tasks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1658–1667, 2019.

[14] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, pp. 1057–1063, 2000.

[15] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust region policy

optimization," *32nd International Conference on Machine Learning, ICML 2015*, vol. 3, pp. 1889–1897, 2015.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *ArXiv*, vol. abs/1707.0, 2017.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–12, 2016.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *35th International Conference on Machine Learning, ICML 2018*, pp. 2976–2989, 2018.

[19] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, pp. 5049–5059, 2017.

[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–21, 2016.

[21] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards," *arXiv preprint*, vol. abs/1707.0, 2017.

[22] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming Exploration in Reinforcement Learning with Demonstrations," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 6292–6299, 2018.

[23] Y. Fan, J. Luo, and M. Tomizuka, "A learning framework for high precision industrial assembly," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 811–817, 2019.

[24] D. Du, S. Han, N. Qi, H. B. Ammar, J. Wang, and W. Pan, "Reinforcement Learning for Safe Robot Control using Control Lyapunov Barrier Functions," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 9442–9448, 2023.

[25] H. Lin, Y. Lou, P. Quan, Z. Liang, D. Wei, and S. Di, "Small-Scale Zero-Shot Collision Localization for Robots Using RL-CNN," *Applied Sciences*, vol. 13, no. 7, 2023.

[26] T. A. Kessler Faulkner, E. Schaertl Short, and A. L. Thomaz, "Interactive Reinforcement Learning with Inaccurate Feedback," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7498–7504, 2020.

[27] S. Chen, J. Gao, S. Reddy, G. Berseth, A. D. Dragan, and S. Levine, "ASHA: Assistive Teleoperation via Human-in-the-Loop Reinforcement Learning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 7505–7512, 2022.

[28] A. Hiranaka, M. Hwang, S. Lee, C. Wang, L. Fei-Fei, J. Wu, and R. Zhang, "Primitive Skill-Based Robot Learning from Human Evaluative Feedback," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7817–7824, 2023.

[29] Z. Li, K. Xu, L. Liu, L. Li, D. Ye, and P. Zhao, "Provable Offline Rein- forcement Learning with Human Feedback," in *ICML 2023 Worshop on Interactive Learning with human feedback*, pp. 1–19, 2023.

[30] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent Advances

in Robot Learning from Demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 1–34, 2020.

[31] D. S. Brown and S. Niekum, "Toward probabilistic safety bounds for robot learning from demonstration," in *AAAI Fall Symposium - Technical Report*, pp. 10–18, 2017.

[32] S. Calinon, F. Guenter, and A. Billard, "On Learning the Statistical Representation of a Task and Generalizing it to Various Contexts," *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, pp. 2978–2983, 2006.

[33] A. Olivares-Alarcos, S. Foix, and G. Alenyà, "On inferring intentions in shared tasks for industrial collaborative robots," *Electronics*, vol. 8, no. 11, pp. 1–22, 2019.

[34] S. Calinon, P. Evrard, E. Gribovskaya, A. Billard, and A. Kheddar, "Learning collaborative manipulation tasks by demonstration using a haptic interface," in *2009 International Conference on Advanced Robotics, ICAR 2009*, pp. 1–6, IEEE, 2009.

[35] E. Rosen, D. Whitney, E. Phillips, D. Ullman, and S. Tellex, "Testing Robot Teleoperation using a Virtual Reality Interface with ROS Reality," 2018.

[36] J. Spranger, R. Buzatoiu, A. Polydoros, L. Nalpantidis, and E. Boukas, "Human-Machine Interface for Remote Training of Robot Tasks," in *IEEE International Conference on Imaging Systems and Techniques - IST2018*, pp. 1–5, 2018.

[37] D. Vogt, S. Stepputtis, S. Grehl, B. Jung, and H. Ben Amor, "A system for learning continuous human-robot interactions from human-human demonstrations," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2882–2889, 2017.

[38] L. Rozo, S. Calinon, D. G. Caldwell, P. Jiménez, and C. Torras, "Learning Physical Collaborative Robot Behaviors From Human Demonstrations," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 513–527, 2016.

[39] R. Toris, D. Kent, and S. Chernova, "Unsupervised learning of multi-hypothesized pick-and-place task templates via crowdsourcing," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4504–4510, IEEE, 2015.

[40] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 109–116, 2004.

[41] L. Rozo, S. Calinon, D. Caldwell, P. Jim, C. Torras, and I. D. Rob, "Learning Collaborative Impedance-Based Robot Behaviors ," *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence Learning*, pp. 1422–1428, 2013.

[42] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[43] H. B. Amor, G. Neumann, S. Kamthe, O. Kroemer, and J. Peters, "Interaction Primitives for Human-Robot Cooperation Tasks," *IEEE International Conference on Robotics Automation (ICRA)*, pp. 2831–2837, 2014.

[44] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[45] Y. Q. Wang, Y. D. Hu, S. E. Zaatari, W. D. Li, and Y. Zhou, "Optimised Learning from Demonstrations for Collaborative Robots," *Robotics and Computer-Integrated Manufacturing*, vol. 71, no. 102169, 2021.

[46] S. Krishnan, A. Garg, R. Liaw, B. Thananjeyan, L. Miller, F. T. Pokorny, and K. Goldberg, "SWIRL: A sequential windowed inverse reinforcement learning algorithm for robot tasks with delayed rewards," *International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 126–145, 2019.

[47] X. Zhang, L. Sun, Z. Kuang, and M. Tomizuka, "Learning Variable Impedance Control via Inverse Reinforcement Learning for Force-Related Tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2225–2232, 2021.

[48] Z. Zhang, J. Hong, A. S. Enayati, and H. Najjaran, "Using Implicit Behavior Cloning and Dynamic Movement Primitive to Facilitate Reinforcement Learning for Robot Motion Planning," *arXiv preprint*, vol. arXiv:2307, pp. 1–18, 2023.

[49] M. Gharbi, R. Lallement, and R. Alami, "Combining symbolic and geometric planning to synthesize human-aware plans: Toward more efficient combined search.," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 6360–6365, IEEE, 2015.

[50] A. Orthey, M. Toussaint, and N. Jetchev, "Optimizing motion primitives to make symbolic models more predictive," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2868–2873, 2013.

[51] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1503–1510, 2015.

[52] R. Lioutikov, G. Maeda, F. Veiga, K. Kersting, and J. Peters, "Inducing Probabilistic Context-Free Grammars for the Sequencing of Movement Primitives," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5651–5658, 2018.

[53] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[54] S. Paul, J. van Baar, and A. K. Roy-Chowdhury, "Learning from trajectories via subgoal discovery," in *Advances in Neural Information Processing Systems*, pp. 1–11, 2019.

[55] T. Jurgenson, E. Groshev, and A. Tamar, "Sub-Goal Trees – a Framework for Goal-Directed Trajectory Prediction and Optimization," in *Proceedings of the 37th International Conference on Machine Learnin*, pp. 5020–5030, 2020.

[56] X. Pan and Y. Shen, "Human-interactive subgoal supervision for efficient inverse reinforcement learning," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pp. 1380–1387, 2018.

[57] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1470–1477, 2011.

[58] E. Erdem, V. Patoglu, and P. Schüller, "A systematic analysis of levels of integration between high-level task planning and low-level feasibility checks," in *AI Communications*, vol. 29, (NLD), pp. 319–349, IOS Press, jan 2016.

[59] A. E. Gerevini, "An Introduction to the Planning Domain Definition Language (PDDL): Book review," *Artificial Intelligence*, vol. 280, no. 2, pp. 1–187, 2020.

[60] D. Holler, G. Behnke, P. Bercher, S. Biundo, H. Fiorino, D. Pellier, and R. Alford, "HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems," *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pp. 9883–9891, 2020.

[61] R. Zhang, J. Li, P. Zheng, Y. Lu, J. Bao, and X. Sun, "A fusion-based spiking neural network approach for predicting collaboration request in human-robot collaboration," *Robotics and Computer-Integrated Manufacturing*, vol. 78, no. May, p. 102383, 2022.

[62] R. Zhang, X. Li, Y. Zheng, J. Lv, J. Li, P. Zheng, and J. Bao, "Cognition-driven Robot Decision Making Method in Human-robot Collaboration Environment," in *IEEE International Conference on Automation Science and Engineering*, pp. 54–59, IEEE, 2022.

[63] K. Darvish, E. Simetti, F. Mastrogiovanni, and G. Casalino, "A hierarchical architecture for human–robot cooperation processes," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 567–586, 2021.

[64] L. S. de Mello and A. C. Sanderson, "AND/OR graph representation of assembly plans," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 2, pp. 188–199, 1990.

[65] E. C. Grigore, A. Roncone, O. Mangin, and B. Scassellati, "Preference-Based Assistance Prediction for Human-Robot Collaboration Tasks," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4441–4448, 2018.

[66] B. Hayes and B. Scassellati, "Autonomously constructing hierarchical task networks for planning and human-robot collaboration," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5469–5476, 2016.

[67] Y. Cheng, L. Sun, and M. Tomizuka, "Human-aware robot task planning based on a hierarchical task model," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1136–1143, 2021.

[68] C. Yu, Y. Ji, G. Qi, X. Gu, and L. Tao, "Group-based production scheduling for make-to-order production," *Journal of Intelligent Manufacturing*, vol. 26, no. 3, pp. 585–600, 2015.

[69] S. Pirk, K. Hausman, A. Toshev, and M. Khansari, "Modeling Long-horizon Tasks as Sequential Interaction Landscapes," in *Proceedings of Machine Learning Research*, vol. 155, pp. 471–484, 2020.

[70] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[71] Z. Su, O. Kroemer, G. E. Loeb, G. S. Sukhatme, and S. Schaal, "Learning manipulation graphs from demonstrations using multimodal sensory signals," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2758–2765, 2018.

[72] D. A. Huang, S. Nair, D. Xu, Y. Zhu, A. Garg, L. Fei-Fei, S. Savarese, and J. C. Niebles, "Neural task graphs: Generalizing to unseen tasks from a single video demonstration," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8557–8566, 2019.

[73] R. Li, A. Jabri, T. Darrell, and P. Agrawal, "Towards Practical Multi-Object Manipulation using Relational Reinforcement Learning," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4051–4058, 2020.

[74] Y. Lin, A. S. Wang, E. Undersander, and A. Rai, "Efficient and Interpretable Robot Manipulation with Graph Neural Networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2740–2747, 2022.

[75] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani, "Object-centric Forward Modeling for Model Predictive Control," in *Conference on Robot Learning (CoRL)*, pp. 100–109, PMLR, 2019.

[76] P. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," *Advances in Neural Information Processing Systems*, pp. 4509–4517, 2016.

[77] T. Silver, R. Chitnis, A. Curtis, J. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, "Planning with Learned Object Importance in Large Problem Instances using Graph Neural Networks," in *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, pp. 11962–11971, 2021.

[78] F. D. Felice, S. D'Avella, A. Remus, P. Tripicchio, and C. A. Avizzano, "One-shot Imitation Learning with Graph Neural Networks for Pick-and-Place Manipulation Tasks," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5926–5933, 2023.

[79] K. Hori, K. Suzuki, and T. Ogata, "Interactively Robot Action Planning with Uncertainty Analysis and Active Questioning by Large Language Model," in *2024 IEEE/SICE International Symposium on System Integration, SII 2024*, pp. 85–91, IEEE, 2024.

[80] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, Z. Xu, D. Sadigh, A. Zeng, and A. Majumdar, "Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners," in *Proceedings of Machine Learning Research*, vol. 229, pp. 1–24, 2023.

[81] H. Zhou, G. Yang, B. Wang, X. Li, R. Wang, X. Huang, H. Wu, and X. V. Wang, "An attention-based deep learning approach for inertial motion recognition and estimation in human-robot collaboration," *Journal of Manufacturing Systems*, vol. 67, no. January, pp. 97–110, 2023.

[82] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Perez, "Integrated Task and Motion Planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 265–293, 2021.

[83] M. Mansouri, F. Pecora, and P. Schüller, "Combining Task and Motion Planning: Challenges and Guidelines," *Frontiers in Robotics and AI*, vol. 8, no. May, pp. 1–12, 2021.

[84] K. Fang, Y. Zhu, A. Garg, S. Savarese, and L. Fei-Fei, "Dynamics Learning with Cascaded Variational Inference for Multi-Step Manipulation," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 1–13, IEEE, jun 2020.

[85] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Sampling-based methods for factored task and motion planning," *International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1796–1825, 2018.

[86] J. Kurosu, A. Yorozu, and M. Takahashi, "Simultaneous dual-arm motion planning for minimizing operation time," *Applied Sciences*, vol. 7, no. 12, pp. 1–14, 2017.

[87] J. K. Behrens, R. Lange, and M. Mansouri, "A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 8705–8711, IEEE, 2019.

[88] H. Liu and L. Wang, "Gesture recognition for human-robot collaboration: A review," *International Journal of Industrial Ergonomics*, vol. 68, pp. 355–367, 2018.

[89] L. Brèthes, P. Menezes, F. Lerasle, and J. Hayet, "Face tracking and hand gesture recognition for human-robot interaction," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1901–1906, 2004.

[90] H. Bay, T. Tuytelaars, and L. V. Gool, "LNCS 3951 - SURF: Speeded Up Robust Features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006.

[91] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571, 2011.

[92] Y. Chen, W. Wang, V. Krovi, and Y. Jia, "Enabling robot to assist human in collaborative assembly using convolutional neural networks," *IEEE International Conference on Intelligent Robots and Systems*, pp. 11167–11172, 2020.

[93] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 142–149, 2000.

[94] M. S. Mohd and S. A. Suandi, "Hand gesture tracking system using Adaptive Kalman Filter," in *Proceedings of the 2010 10th International Conference on Intelligent Systems Design and Applications, ISDA'10*, pp. 166–171, 2010.

[95] G. Du and P. Zhang, "A Markerless Human-Robot Interface Using Particle Filter and Kalman Filter for Dual Robots," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2257–2264, 2015.

[96] D. Wu and L. Shao, "Leveraging Hierarchical Parametric Networks for Skeletal Joints Based Action Segmentation and Recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 724–731, 2014.

[97] Y. Cheng, F. Sun, Y. Zhang, and F. Tao, "Task allocation in manufacturing: A review," *Journal of Industrial Information Integration*, vol. 15, no. June 2018, pp. 207–218, 2019.

[98] D. Moutinho, L. F. Rocha, C. M. Costa, L. F. Teixeira, and G. Veiga, "Deep learning-based human action recognition to leverage context awareness in collaborative assembly," *Robotics and Computer-Integrated Manufacturing*, vol. 80, no. October, p. 102449, 2023.

[99] P. Rückert, B. Papenberg, and K. Tracht, "Classification of assembly operations using machine learning algorithms based on visual sensor data," *Procedia CIRP*, vol. 97, pp. 110–116, 2020.

[100] S. Franklin, T. Madl, S. Strain, U. Faghihi, D. Dong, S. Kugele, J. Snaider, P. Agrawal, and S. Chen, "A LIDA cognitive model tutorial," *Biologically Inspired Cognitive Architectures*, vol. 16, pp. 105–130, 2016.

[101] J. R. Anderson, "ACT A Simple Theory of Complex Cognition," *Carnegie Mellon University*, vol. 51, no. 4, pp. 355–365, 1996.

[102] F. E. Ritter, F. Tehranchi, and J. D. Oury, "ACT-R: A cognitive architecture for modeling cognition," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 10, no. 3, 2019.

[103] J. Oyekan, Y. Chen, C. Turner, and A. Tiwari, "Applying a fusion of wearable sensors and a cognitive inspired architecture to real-time ergonomics analysis of manual assembly tasks," *Journal of Manufacturing Systems*, vol. 61, pp. 391–405, 2021.

[104] I. Kotseruba and J. K. Tsotsos, "A Review of 40 Years of Cognitive Architecture Research: Core Cognitive Abilities and Practical Applications," *arXiv preprint*, vol. abs/1610.0, 2016.

[105] L. Bruckschen, K. Bungert, N. Dengler, and M. Bennewitz, "Predicting human navigation goals based on Bayesian inference and activity regions," *Robotics and Autonomous Systems*, vol. 134, p. 103664, 2020.

[106] D. Morrison, P. Corke, and J. Leitner, "Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach," in *Robotics: Science and Systems*, pp. 1–10, 2018.

[107] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International Journal of Robotics Research*, vol. 39, pp. 183–201, jun 2019.

[108] M. Sundermeyer, R. Triebel, and R. O. Mar, "Contact-GraspNet : Efficient 6-DoF Grasp Generation in Cluttered Scenes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[109] Z. Zhang, C. Zhou, Y. Koike, and J. Li, "Single RGB Image 6D Object Grasping System Using Pixel-Wise Voting Network," *Micromachines*, vol. 13, no. 2, pp. 1–13, 2022.

[110] Y. Huang, A. Conkey, and T. Hermans, "Planning for Multi-Object Manipulation with Graph Neural Network Relational Classifiers," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1822–1829, IEEE, 2023.

[111] L. Wang, X. Meng, Y. Xiang, and D. Fox, "Hierarchical Policies for Cluttered-Scene Grasping with Latent Plans," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2883–2890, 2022.

[112] Y. Li, W. Ouyang, X. Wang, and X. Tang, "ViP-CNN: Visual phrase guided convolutional neural network," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 7244–7253, 2017.

[113] S. Qi, W. Wang, B. Jia, J. Shen, and S. C. Zhu, "Learning human-object interactions by graph parsing neural networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11213 LNCS, pp. 407–423, 2018.

[114] Y. Lu, C. Chang, H. Rai, G. Yu, and M. Volkovs, "Learning Effective Visual Relationship Detector on 1 GPU," in *International Conference on Computer Vision*, pp. 1–8, 2019.

[115] Y. Lu, H. Rai, J. Chang, B. Knyazev, G. Yu, S. Shekhar, G. W. Taylor, and M. Volkovs, "Context-aware Scene Graph Generation with Seq2Seq Transform-

ers," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 15911–15921, 2021.

[116] M. Guermal, R. Dai, and F. Bremond, "THORN: Temporal Human-Object Relation Network for Action Recognition," in *Proceedings - International Conference on Pattern Recognition*, pp. 3303–3309, 2022.

[117] W. Goodwin, S. Vaze, I. Havoutis, and I. Posner, "Semantically Grounded Object Matching for Robust Robotic Scene Rearrangement," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 11138–11144, 2022.

[118] P. Ardon, E. Pairet, R. P. Petrick, S. Ramamoorthy, and K. S. Lohan, "Learning Grasp Affordance Reasoning through Semantic Relations," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4571–4578, 2019.

[119] H. Zhang, X. Lan, S. Bai, X. Zhou, Z. Tian, and N. Zheng, "ROI-based Robotic Grasp Detection for Object Overlapping Scenes," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4768–4775, 2019.

[120] R. Zhang, F. Torabi, L. Guan, D. H. Ballard, and P. Stone, "Leveraging human guidance for deep reinforcement learning tasks," in *IJCAI International Joint Conference on Artificial Intelligence*, pp. 6339–6346, 2019.

[121] G. Zuo, J. Tong, H. Liu, W. Chen, and J. Li, "Graph-Based Visual Manipulation Relationship Reasoning Network for Robotic Grasping," *Frontiers in Neurorobotics*, vol. 15, no. August, pp. 1–12, 2021.

[122] G. Zuo, J. Tong, H. Liu, W. Chen, and J. Li, "Graph-based Visual Manipulation Relationship Reasoning in Object-Stacking Scenes," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–8, 2021.

[123] Y. Torres, S. Nadeau, and K. Landau, "Classification and quantification of human error in manufacturing: A case study in complex manual assembly," *Applied Sciences*, vol. 11, no. 2, pp. 1–23, 2021.

[124] W. Wang, Y. Chen, R. Li, and Y. Jia, "Learning and Comfort in Human–Robot Interaction: A Review," *Applied Sciences*, vol. 9, no. 23, 2019.

[125] A. Muxfeldt and J. J. Steil, "Recovering from Assembly Errors by Exploiting Human Demonstrations," *Procedia CIRP*, vol. 72, pp. 63–68, 2018.

[126] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 7579–7586, 2018.

[127] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pp. 1–14, 2014.

[128] H. Kim, N. Yoshimura, and Y. Koike, "Characteristics of Kinematic Parameters in Decoding Intended Reaching Movements Using Electroencephalography (EEG)," *Frontiers in Neuroscience*, vol. 13, p. 1148, 2019.

[129] H. M. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé, "Hierarchical imitation and reinforcement learning," in *35th International Conference on Machine Learning, ICML 2018*, pp. 4560–4573, PMLR, 2018.

[130] T. de Bruin, J. Kober, K. Tuyls, and R. Babuska, "The importance of experience replay database composition in deep reinforcement learning," in *Deep Reinforcement Learning Workshop, Advances in Neural Information Processing Systems (NIPS)*, pp. 1–9, 2015.

[131] E. Rohmer, S. P. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 1321–1326, IEEE, 2013.

[132] R. Killick, P. Fearnhead, and I. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, pp. 1590–1598, 2012.

[133] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, pp. 1025–1035, 2017.

[134] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, pp. 1–9, 2019.

[135] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems*, vol. 32, pp. 1–13, 2019.

[136] J. Wang, "An intuitive tutorial to Gaussian processes regression," *arXiv preprint arXiv:2009.10862*, 2020.

[137] V. Joukov and D. Kulic, "Gaussian process based model predictive controller for imitation learning," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 850–855, 2017.

[138] J. J. Kuffner and S. M. La Valle, "RRT-connect: an efficient approach to single-query path planning," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 995–1001, 2000.

[139] I. Sucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library (OMPL)," *IEEE Robotics Automation Magazine*, no. December, pp. 1–10, 2010.

[140] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint*, vol. abs/1804.0, 2018.

[141] M. Fleder, *Robot Operating System (ROS) The Complete Reference (Volume 1)*, vol. 1. Springer Publishing Company, Incorporated, 1st ed., 2012.

[142] M. Gorner, R. Haschke, H. Ritter, and J. Zhang, "Moveit! task constructor for task-level motion planning," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 190–196, 2019.

[143] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A Framework for Building Perception Pipelines," *ArXiv*, vol. abs/1906.0, pp. 1–9, 2019.

[144] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM Fully Convolutional Networks for Time Series Classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2017.

[145] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015*, pp. 1–14, 2015.

[146] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-Term Recurrent Convolutional Networks for Visual Recognition and Description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017.

[147] T. Chen, A. Shenoy, A. Kolinko, S. Shah, and Y. Sun, "Multi-Object Grasping-Estimating the Number of Objects in a Robotic Grasp," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 4995–5001, 2021.

[148] F. Abi-Farraj, C. Pacchierotti, O. Arenz, G. Neumann, and P. R. Giordano, "A Haptic Shared-Control Architecture for Guided Multi-Target Robotic Grasping," *IEEE Transactions on Haptics*, vol. 13, no. 2, pp. 270–285, 2020.

[149] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[150] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pp. 4602–4609, 2019.

[151] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, pp. 5999–6009, 2017.

[152] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." https://github.com/facebookresearch/detectron2, 2019.

[153] H. Zhang, X. Lan, X. Zhou, Z. Tian, Y. Zhang, and N. Zheng, "Visual manipulation relationship recognition in object-stacking scenes," *Pattern Recognition Letters*, vol. 140, pp. 34–42, 2020.

[154] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas, "A Generalist Agent," *Transactions on Machine Learning Research*, vol. 1, no. 1, pp. 1–42, 2022.

[155] C. Huang, C. Chan, C. Pan, C. Fu, C. Devin, D. Driess, D. Pathak, D. Shah, and B. Dieter, "Open X-Embodiment : Robotic Learning Datasets and RT-X Models," *arXiv preprint*, vol. arXiv:2310, pp. 1–12, 2023.