RIGA TECHNICAL UNIVERSITY

Jānis Ārents

RESEARCH AND DEVELOPMENT OF SMART CONTROL METHODS FOR INDUSTRIAL ROBOTS

Doctoral Thesis

RTU Press
Riga 2023

RIGA TECHNICAL UNIVERSITY
Faculty of Electrical and Environmental Engineering
Institute of Industrial Electronics and Electrical Engineering

Jānis Ārents

Doctoral Student of the Study Programme "Computerised Control of Electrical Technologies"

RESEARCH AND DEVELOPMENT OF SMART CONTROL METHODS FOR INDUSTRIAL ROBOTS

Doctoral Thesis

Scientific supervisor
Director, Senior researcher, Dr. sc. comp.
MODRIS GREITĀNS
Institute of Electronics and Computer Science

Professor, Senior researcher, Dr. sc. ing.
PĒTERIS APSE-APSĪTIS
Riga Technical university

Riga 2023

ABSTRACT

Smart industrial robots are not performing ordinary tasks, where the situation is predetermined and the environment known beforehand. Smart industrial robots are subjected to external conditions that must be acquired, interpreted and reacted to. Several significant challenges are limiting the robots of today to achieving a higher degree of such characteristics as functional suitability, usability and maintainability. The Thesis addresses the smart control methods for industrial robots, their applications, and learning strategies. Particularly efficient data preparation, computer-vision-based robot control, and knowledge acquisition from humans through demonstrations. The primary aim of the Thesis is to research and develop novel techniques in these fields and, therefore, improve the potential deployment of industrial robots for working in dynamic environments and advance the operability of such systems.

The literature review leads to a formalization of the smart industrial robot concept and discusses different control strategies and their importance within the core functionalities of smart industrial robot control. Based on the identified challenges and open issues during the literature review, the Thesis develops novel synthetic data generation techniques for usage in computer-vision-based robot control and an approach for learning from human demonstrations.

To decrease the required manual work for the data-gathering process and increase the precision of object detection tasks, a new method of synthetic data generation is presented and described in detail, including the relevant tests and comparisons of deep-learning-based object detection models trained on various combinations of synthetic and real datasets. Computer-vision-based robot control, however, builds upon the synthetic data generation approaches and shows the viability of the usage in robotic systems. The experimental results of imitation learning show the viability of using human demonstrations within the smart robot control and, therefore, easing the required expertise in the process of robot programming. An experimental setup is built where the methods are demonstrated, consisting of various use cases, tackling important challenges of today's manufacturing.

The doctoral Thesis proves three statements and as a result, efficient and smart control methods for industrial robot control are researched and developed. It is a result of research carried out at the Institute of Electronics and Computer Science (EDI) within several national and EU projects. It consists of 125 pages, 57 figures, 6 tables, 166 sources of literature and two appendices.

ANOTĀCIJA

Viedie industriālie roboti neveic parastus uzdevumus, kur situācija ir iepriekš noteikta un vide zināma. Viedie industriālie roboti ir pakļauti ārējiem apstākļiem, kas ir jāuztver, jāinterpretē un uz tiem jāreaģē. Vairāki nozīmīgi izaicinājumi ierobežo mūsdienu robotus, lai sasniegtu augstāku tādu īpašību pakāpi kā funkcionālā piemērotība, lietojamība un pielāgojamība. Šajā darbā apskatītas industriālo robotu viedās vadības metodes, to pielietojumi un mācīšanās stratēģijas. Savukārt galvenais fokuss tiek atvēlēts efektīvai datu sagatavošanai, uz datorredzes balstītai robotu vadībai un cilvēku demonstrāciju izmantošanai industriālo robotu vadībā. Šī promocijas darba mērķis ir izpētīt un izstrādāt jaunas tehnikas šajās jomās un tādējādi uzlabot industriālo robotu potenciālo izmantošanu darbam dinamiskā vidē, kā arī uzlabot vairākus nozīmīgus šādu sistēmu darbības aspektus.

Literatūras apskats noved pie viedo industriālo robotu koncepta formalizēšanas un apspriež dažādas vadības stratēģijas un to nozīmi viedo rūpniecisko robotu vadības pamatfunkcijās. Balstoties uz literatūras apskatā identificētajiem izaicinājumiem, promocijas darbā izstrādātas jaunas sintētisko datu ģenerēšanas metodes izmantošanai uz datorredzi balstītā robotu vadībā un pieeja cilvēku demonstrāciju izmantošanai robotu vadība.

Lai samazinātu nepieciešamo manuālo darbu datu ieguves procesā un palielinātu objektu noteikšanas uzdevumu precizitāti, tiek prezentēta un detalizēti aprakstīta jauna sintētisko datu ģenerēšanas metode, kas ietver attiecīgus testus un uz dziļās mašīnmācīšanās balstītu objektu noteikšanas modeļu salīdzinājumus, kur modeļi apmācīti uz dažādām sintētisko un reālo datu kopu kombinācijām. Uz datorredzi balstītā robotu vadība izmanto sintētisko datu ģenerēšanas pieejas un demonstrē pielietojumu robotizētās sistēmās. Atdarinošās mašīnmācīšanās eksperimentālie rezultāti parāda cilvēku demonstrāciju pielietojamību viedo robotu vadībā un tādējādi atvieglo nepieciešamās zināšanas robotu programmēšanas procesā. Tiek izveidota eksperimentālā sistēma, kurā demonstrētas metodes dažādos pielietojumos, risinot svarīgus izaicinājumus mūsdienu ražošanas apstākļos.

Promocijas darbs pierāda trīs apgalvojumus un rezultātā tiek izstrādātas efektīvas un viedas industriālo robotu vadības metodes. Disertācija ir izstrādāta Elektronikas un datorzinātņu institūtā (EDI) vairāku valsts un ES projektu ietvaros. Darbs sastāv no 125 lappusēm, 57 attēliem, sešām tabulām, 166 literatūras avotiem un diviem pielikumiem.

ACKNOWLEDGEMENTS

I am deeply grateful to my supervisor, Modris Greitāns, whose unwavering guidance, expertise, and encouragement have been instrumental in shaping this research and my academic growth. Your insightful feedback and patient mentoring have been invaluable to me, and I am truly fortunate to have had the opportunity to learn under your guidance. I am also thankful to Pēteris Apse-Apsītis for guiding me through this long journey.

I would also like to extend my heartfelt appreciation to my colleagues from EDI, especially the robotics group, RuM and CPS laboratories. Our collaborative discussions, exchange of ideas, and shared challenges have enriched my research experience and broadened my perspectives.

To my beloved wife, Dina, and our daughter, Evelīna, your unwavering support, love, and understanding have been my pillars of strength. Your patience during the long hours and your belief in me have made this journey possible. I am grateful beyond words for your sacrifices and encouragement.

I am thankful to my family and friends for their constant encouragement and support and to all those who have contributed in ways seen and unseen.

Jānis Ārents
2023

CONTENTS

List of Figures

List of Tables

ABBREVIATIONS

2D – 2 Dimensional
3D – 3 Dimensional
ABB – Asea Brown Boveri
ACM – Allowed Collision Matrix
ANN – Artificial Neural Network
AP – Average Precision
CAD – Computer-aided design
CHOMP – Covariant Hamiltonian Optimization for Motion Planning
CNN – Convolutional Neural Network
CPU – Central Processing Unit
CV – Computer Vision
DL – Deep Learning
DMP – Dynamic Motion Primitive
DOF – Degrees of Freedom
EDI – Institute of Electronics and Computer Science
EOAT – End of Arm Tooling
FAST – Features from Accelerated Segment Test
FCL – Flexible Collision Library
FCN – Fully Convolutional Network
FN – False Negative
FP – False Positive
FPS – Frames Per Second
GAN – Generative Adversarial Network
GPU – Graphics Processing Unit
GRU – Gated Recurrent Unit
GUI – Graphical User Interface
HG – Hand Guiding
HRC – Human-Robot Collaboration
HRI – Human-Robot Interaction
IK – Inverse Kinematics
IoU – Intersection over Union
IRL –Inverse Reinforcement Learning
KDL – Kinematics and Dynamics Library
LSTM – Long Short-Term Memory
mAP – mean Average Precision
MDP – Markov Decision Process

OMPL – Open Motion Planning Library
PCA – Principal Component Analysis
PFL – Power and Force Limiting
PRM – Probalistic Roadmap
RAM – Random-Access Memory
R-CNN – Regions with CNN features
RGB – Red, Green, Blue
RGB-D – Red, Green, Blue, Depth
ReLU – Rectified Linear Unit
RL – Reinforcement Learning
RNN – Recurrent Neural Network
ROI – Region of Interest
ROS – Robot Operating System
ROS-I – ROS-Industrial
Rviz – ROS visualization
SRDF – Semantic Robot Description Format
SRMS – Safety-Rated Monitored Stop
SSM – Speed and Separation Monitoring
SVM – Support Vector Machine
STOMP – Stochastic Trajectory Optimization for Motion Planning
TCP – Tool Center Point
TP – True Positive
URDF – Unified Robot Description Format
VR – Virtual Reality
Xacro – XML macro language
XML – eXtensible Markup Language
YOLO – You Only Look Once

NOMENCLATURE

π, π_θ – policy, policy with parameters θ
s – state of the system or direct observation
s_t, s' – state in time step t, future s
a, a_t – action, action in time step t
o – observation
$R(s), R(s, a)$ – reward function
θ, ϕ, ψ – model parameter vectors
$\sigma(x)$ – neural network activation function
$\boldsymbol{x}, x_i$ – vector, element of vector
$\mathbb{R}^n$, $[0, 1]^n$ – n-dimensional vector sets
$\boldsymbol{r}$ – translation vector
g_t – gripper actuator signal
q – quaternion
$\mathcal{D}, \tau \in \mathcal{D}$ – demonstration set, trajectory
γ – discount factor

INTRODUCTION

Industrial robots and their control methods have come a long way since their first appearance in manufacturing [1]. The essence of industrial robots is to automate repetitive, dangerous, heavy-duty, and resource-intensive tasks, etc., and eventually to automate human-intelligence-demanding tasks. An industrial robot is defined by ISO 8373:2012 as follows: "An automatically controlled, reprogrammable, multi-purpose manipulator programmable in three or more axes, which can be either fixed in place or mobile for use in industrial automation applications" [2]. The core element of an industrial robot is the combination of motors that drives the robot and control algorithms. A precise sequence of executed actions enables an industrial robot to achieve desirable trajectories and succeed at the given tasks. There are several different mechanical structures of industrial robots [3], and the choice of robot structure depends on space and the task itself. Due to its universality, the most commonly used industrial robots have articulated mechanical design that usually consists of at least six degrees of freedom (DOF).

The third industrial revolution proved to the industry and society how the usage of industrial robots is helping manufacturing companies to become more effective and competitive in their field [4]. Since their first appearance, industrial robots have been applied to automate manufacturing processes on many production floors, therefore transforming the sector of manufacturing. Similarities can be drawn today in modern industry in the era of digitization, in a time when Industry 4.0 [5] concepts are being applied, and industrial robots with more advanced capabilities are one of the core elements of the transformation process and enablers of smart manufacturing.

Substantial technological developments, new applications, global trends in rising labour costs, and population ageing are only some of the aspects in the growth of annual shipments of industrial robots. In fact, from 2016 to 2022, annual installations of industrial robots increased by 14 % on average each year [6]. However, currently, the majority of industrial robots that are installed in production floors have been traditionally programmed to meet the specific needs of respective factories and mostly tailored to perform "dull, dirty, or dangerous" jobs [7]. Typically, such industrial robot systems are neither intelligent nor able to adapt to the dynamic environment nor learn tasks by themselves. Therefore, a significant amount of time is spent programming industrial robots [8] by hand-crafting trajectories for one specific task, and it is incredibly laborious, complicated and error-prone. Thus, the environment must be deterministic, and everything must be in place and precisely positioned. If the design of the product changes, the programming must be performed all over again. Changes in the marketplace translate into uncertainty for manufacturing and end-user mobility services. Customers today want new models or versions of products faster than ever before, and the way for businesses to succeed is by being flexible, smart and effective in the manufacturing process.

Factories of today are still effectively designed for a single purpose, which means there is little or no room for flexibility in terms of product design changes. By looking at smart manu-

facturing and digitization trends [9], we see that future factories will be multi-purpose and able to adapt to new designs in a very short amount of time. However, the possibilities that come with digitization in smart manufacturing are not yet fully explored nor exploited, but some of the technologies have already marked their spot for a long-term stay in the manufacturing sector. AI-based approaches have already been internationally accepted as the main driver [10] in a transformation and digitization of factories as flexibility and deep understanding of complex manufacturing processes are becoming the key advantages to raise competitiveness [11]. The smart factory, in a way, is a manufacturing solution driven by smart industrial robots, acting as one of the key elements of Industry 4.0 [12] and as an enabling technology of Industry 5.0 where the creativity of human experts in conjunction with smart, efficient and accurate machines is explored [13].

Smart industrial robots, also referred to as intelligent industrial robots in the literature, have been mentioned as a "remarkably useful combination of a manipulator, sensors and controls" [14]. Even though the need for smart industrial robots and first discussions dates back to the 1980s [15], the idea on a high level remains the same—the wide variety of sensors in combination with reasoning abilities and control mechanisms are used to achieve desired industrial robot motions. This intelligence in robotic systems can be achieved in various ways, therefore also different components, methods and strategies act as enablers of smart industrial robot control. The main component, however, in this ecosystem can be considered a perception and the learning process as a common intelligence support mechanism. Perception is understood as a system that provides the robots with perceiving, comprehending and reasoning abilities. This includes sensory data processing, representation and interpretation [16]. Within the functionality of perception, several key elements can be distinguished that fundamentally differ the learning strategies of robots. These components are mainly associated with the way knowledge is acquired, generated by different strategies and applied in context-aware task execution. Essentially, the knowledge and database is the core element that enables smart industrial robot control. Acquiring new knowledge and generalizing that knowledge enables one to undertake new tasks based on the history of decisions.

The Thesis addresses the smart control methods for industrial robots, their applications, and learning strategies. Particularly efficient data preparation, computer-vision-based robot control, and knowledge acquisition from humans through demonstrations. **The primary aim of the Thesis is to research and develop novel techniques in these fields and therefore improve the potential deployment of industrial robots for working in dynamic environments and advance the operability of such systems.**

Smart industrial robots are not performing ordinary tasks, where the situation is predetermined and the environment known beforehand. Smart industrial robots are subjected to external conditions that must be acquired, interpreted and reacted to. Several significant challenges are limiting the robots of today to achieving a higher degree of such characteristics as functional

suitability, usability and maintainability. Therefore, being more efficient in the training process, more adaptable to the unknown and more intuitive in the interaction with humans, both in the training process and while operating, are some of the key aspects in advancing industrial robot systems. In the end, the smart robots will be as good as the data provided in the training steps, therefore the data preparation and quality of it are also emphasized in the Thesis. However, the precision for some of the proposed methods depends on the properties of the objects used. If not stated otherwise the objects used in experiments are simple-shaped - bilaterally symmetric cylindrical shaped and parallelepipedic shaped objects.

Considering the above-mentioned, several tasks have been defined in order to reach the aim of the Thesis:

- to perform a review of the literature on smart control methods for industrial robots;
- to formalize the concept of smart control of industrial robots;
- to research and develop efficient data preparation methods for computer-vision-based robot control;
- to research and develop a methodology for human demonstration incorporation in the knowledge acquisition process;
- to construct an experimental setup and apply the proposed methods in demonstrations;
- to draw conclusions about the Thesis results.

On the basis of the set tasks, three theses for defence are put forward:

1. The utilization of synthetic data in the training process for simple-shaped object detection in bin-picking setup reduces the required manual effort, whilst trained models are able to detect at least one object in every scene with an IoU threshold above 0.95.
2. For computer-vision-based robot control in the case of simple-shaped objects at least one valid grasp is found in bin-picking setup in more than 99 % of scenes when computer-vision algorithms have been trained solely on synthetic data.
3. The learned trajectories for an object-throwing task by utilizing indirect human demonstrations achieve cosine similarity of more than 0.98 when compared to the validation dataset and, when executed on a real robot, closely resemble the expert human throwing motions.

Taking into account the main goal of the thesis and the statements put forward the thesis is divided into six chapters. Chapter 1 is an overview of smart industrial robot control methods. It begins with a formalization of the smart industrial robot concept, including the identification of

the core functionalities, continues with an overview of different control strategies, and lists the identified challenges and open issues at the end. Chapter 2 shows the developed synthetic data generation techniques and usage in deep-learning-based tasks. Chapter 3 builds upon the synthetic data generation approaches for usage in robotic systems, namely as computer-vision-based robot control. Chapter 4 explores learning from human demonstrations, therefore, imitating humans and realizing imitation-based robot control. Chapter 5 describes the experimental setup and demonstrations where the developed approaches are applied and qualitatively evaluated. The conclusions about the developed methods and test results are presented in Chapter 6.

The results presented in the Thesis are mostly the result of research activities in Horizon 2020 Electronic Components and Systems for European Leadership (ECSEL) projects:

- "Artificial Intelligence for Digitizing Industry" (**AI4DI**[1], GA:826060),
- "Vision, Identification, with Z-sensing Technology and key Applications" (VIZTA[2], GA:826600),
- "Intelligent Motion Control under Industry 4.E" (**IMOCO4.E**[3], GA:101007311).

Horizon 2020 project:

- "Digital Technologies, Advanced Robotics and increased Cyber-security for Agile Production in Future European Manufacturing Ecosystems" (**TRINITY**[4], GA:825196).

Horizon Europe, Key Digital Technologies Joint Undertaking (KDT JU) project:

- "Edge AI Technologies for Optimised Performance Embedded Processing" (**EdgeAI**[5], GA:101097300).

National funding projects:

- Latvian Council of Science, "Smart Materials, Photonics, Technologies and Engineering Ecosystem" project No VPP-EM-FOTONIKA-2022/1-0001, **MOTE**[6],
- Latvian–Lithuanian–Taiwanese scientific cooperation support fund's project "Development of microrobot based on visual recognition and machine learning for manipulation of individual living cells", Nr. LV-LT-TW/2021/8, 2020-2023, **RoVam**[7],

[1] *https://ai4di.eu/*
[2] *https://www.vizta-ecsel.eu/*
[3] *https://www.imoco4e.eu/*
[4] *https://trinityrobotics.eu/*
[5] *https://edge-ai-tech.eu/*
[6] *https://fotonika.lv/*
[7] *https://www.edi.lv/en/projects/development-of-microrobot-based-on-visual-recognition-and-machine-learning-for-manipulation-of-individual-living-cells-rovam/*

- Project No. 1.2.1.1/16/A/002 “Competency centre for Latvian Electrical and optical manufacturing industry” Research no. 11 “The research on the development of computer vision techniques for the automation of industrial processes”, **DIPA**[8].

The results are also presented in several published papers in various scientific journals:

- **Arents, J.**, Cacurs, R., & Greitans, M. (2018). Integration of computer vision and artificial intelligence subsystems with robot operating system-based motion planning for industrial robots. Automatic Control and Computer Sciences, 52, 392-401. [17],

- **Arents, J.**, Abolins, V., Judvaitis, J., Vismanis, O., Oraby, A., & Ozols, K. (2021). Human–robot collaboration trends and safety aspects: A systematic review. Journal of Sensor and Actuator Networks, 10(3), 48. [11],

- **Arents, J.**, & Greitans, M. (2022). Smart industrial robot control trends, challenges and opportunities within manufacturing. Applied Sciences, 12(2), 937. [18],

- Torres, P., **Arents, J.**, Marques, H., & Marques, P. (2022). Bin-Picking Solution for Randomly Placed Automotive Connectors Based on Machine Learning Techniques. Electronics, 11(3), 476. [19],

- Racinskis, P., **Arents, J**., & Greitans, M. (2022). A motion capture and imitation learning-based approach to Robot Control. Applied Sciences, 12(14), 7186. [20],

- Feščenko, V., **Ārents, J**., & Kadiķis, R. (2022). Synthetic Data Generation for Visual Detection of Flattened PET Bottles. Machine Learning and Knowledge Extraction, 5(1), 14-28. [21],

- Vismanis O, **Arents J**, Freivalds K, Ahluwalia V, Ozols K. Robotic System for Post Office Package Handling. Applied Sciences. 2023; 13(13):7643. https://doi.org/10.3390/app13137643. [22]

scientific conferences:

- Buls, E., Kadikis, R., Cacurs, R., & **Ārents, J**. (2019, March). Generation of synthetic training data for object detection in piles. In Eleventh International Conference on Machine Vision (ICMV 2018) (Vol. 11041, pp. 533-540). SPIE. [23],

- **Arents, J.**, Lesser, B., Bizuns, A., Kadikis, R., Buls, E., & Greitans, M. (2022, May). Synthetic Data of Randomly Piled, Similar Objects for Deep Learning-Based Object Detection. In Image Analysis and Processing–ICIAP 2022: 21st International Conference,

[8] *https://www.edi.lv/projects/petijums-par-datorredzes-panemienu-attistibu-industrijas-procesu-norises-automatizacijai-dipa/*

Lecce, Italy, May 23–27, 2022, Proceedings, Part II (pp. 706-717). Cham: Springer International Publishing. [24],

- Duplevska, D., Ivanovs, M., **Arents, J.**, & Kadikis, R. (2022, September). Sim2Real image translation to improve a synthetic dataset for a bin picking task. In 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1-7). IEEE. [25].

and book chapters:

- **Arents, J.**, Greitans, M., & Lesser, B. (2022). Construction of a smart vision-guided robot system for manipulation in a dynamic environment. In Artificial Intelligence for Digitising Industry–Applications (pp. 205-220). River Publishers. [26],
- Urlini, G., **Arents, J.**, & Latella, A. (2022). AI in industrial machinery. In Artificial Intelligence for Digitising Industry–Applications (pp. 179-185). River Publishers. [27].

Part of the research activities of the Thesis are included in the **Smart robot system with advanced vision, sensing, and human gesture understanding capabilities** which was evaluated by the Latvian Academy of Sciences in the annual Science Achievements' Competition as one of the most significant scientific achievements in Latvia in 2022[9].

[9]https://www.lza.lv/en/activities/news/1368-winners-of-the-annual-science-achievements-competition-2022-announced

1. OVERVIEW AND TRENDS OF SMART CONTROL METHODS FOR INDUSTRIAL ROBOTS

Since the first discussions on smart industrial robots, the developments and achievements in sensor fields have grown highly [28]; similarly, the trends in industrial robot control methods have significantly changed. The achievements in the field of AI are contributing highly to smart industrial robot control trends. AI usage in robotic systems is strongly becoming one of the main areas of focus as the manufacturing sector requires increased performance, adaptability to product variations, increased safety, reduced costs, etc. [29]. However, these requirements are neither feasible nor sustainable to be achieved by standard control methods. The methods and techniques of traditional industrial robot control are similar to factories that can be single-purpose [30], meaning that methods that are focused on a single task can hardly be adjustable to different scenarios if adjustable at all. As one of the criteria to succeed in modern manufacturing is to be flexible to changes, industrial robots need algorithms that can adjust to different scenarios with minor modifications that can be performed by the machine operators.

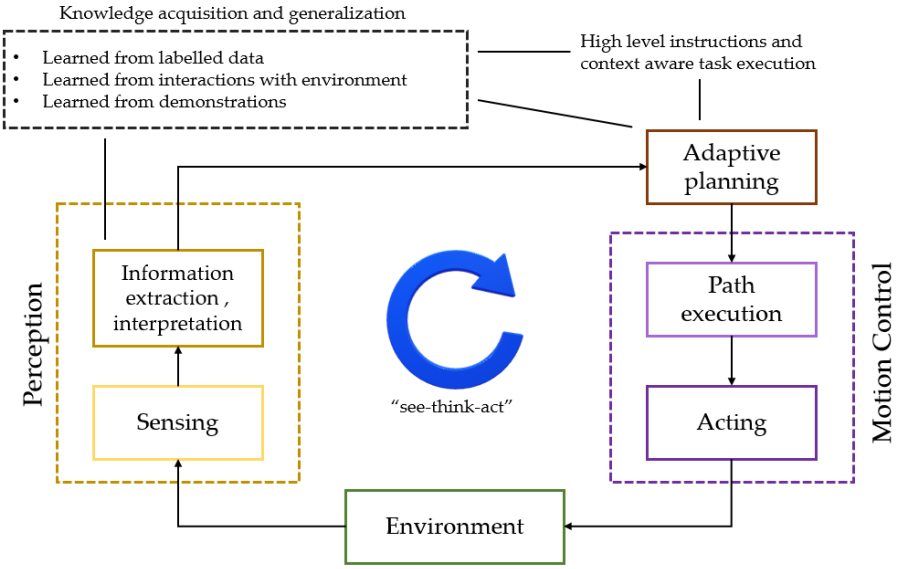
The need for smart industrial robot control methods arises when there is uncertainty in the environment. With the environment in this context, we mainly understand the region in the environment that can be reached by the robot, e.g., the robot's workspace. Typically, this workspace scales down to a region of interest (ROI) or objects of interest to which the robot is manipulating. Nevertheless, there can be different types of uncertainties in the robots' workspace, such as randomly placed objects [17], dynamic obstacles [31], and human presence, etc. Usually, such tasks cannot be preprogrammed, and the robotic system strictly depends on the information that it receives from the environment and the learned strategies. To avoid collisions, succeed in the given tasks and interact with the environment, the usage of additional sensor systems is beneficial. For example, in the field of human-robot collaboration, at least 13 different sensors/devices are used [11]. Additionally, there is also a need to reduce the time spent on programming industrial robots as traditional approaches of handcrafting trajectories [8] is a laborious, complicated and error-prone task.

1.1. Smart industrial robot concept

Smart industrial robot control is an important element, yet only a fraction of the smart manufacturing ecosystem. In the Thesis, and especially in this chapter, the focus is concentrated on an overview of components, methods and strategies that act as enablers of smart industrial robot control. Therefore, only the required core functionalities [32,33] that differ them from traditional industrial robots are investigated and essentially are oriented towards cognitive robotics [34]. The overall concept of smart industrial robot control follows the principle of "see-think-act". Even though this principle originally has been addressed in the field of mobile robotics [35], it

is also becoming more relevant in the context of smart manufacturing. The diagram illustrated in Fig. 1.1 is built upon this principle and integrates such core functionalities of smart industrial robots as:

- Perception;
- High-level instruction and context-aware task execution;
- Knowledge acquisition and generalization;
- Adaptive planning.



1.1. Fig. Smart industrial robot concept [18].

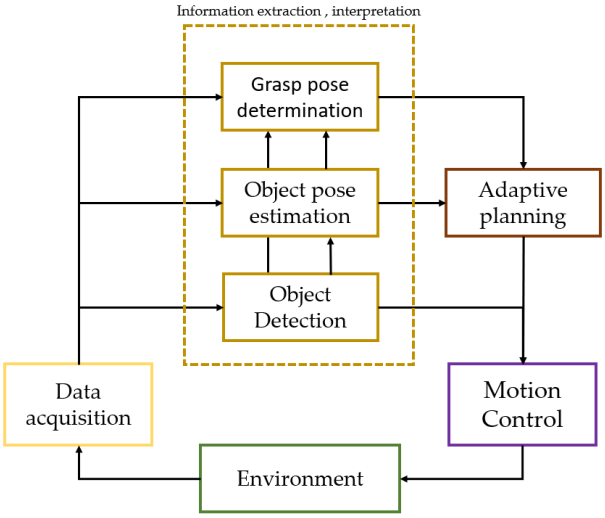
Perception is understood as a system that provides the robots with perceiving, comprehending and reasoning abilities. This includes sensory data processing, representation and interpretation [16]. Within the functionality of perception, several key elements can be distinguished that fundamentally differ from the listed methods and strategies in the following sections of this chapter and Thesis as well. These components are mainly associated with the way how knowledge is acquired, generated by different strategies and applied in context-aware task execution. Essentially, the knowledge and database is the core element that enables smart industrial robot control. Acquiring new knowledge and generalizing that knowledge enables one to undertake new tasks based on the history of decisions. High-level instructions and context-aware task execution, however, factor in application scenario-specific constraints when carrying out the tasks

so robots can determine the best possible actions by themselves to effectively succeed at the given task. The knowledge is supported by adaptive planning that allows one to anticipate events and prepare for them in advance, therefore coping with unforeseen situations [32].

As illustrated in Fig. 1.1, knowledge can be acquired in several different ways. The learning strategies fundamentally differ in how the knowledge is acquired, generalized and essentially how much industrial robot is involved in the learning process. Learning from labelled data in the Thesis is associated with computer-vision-based control and supervised learning, where the current trends are explored in Section 1.2. In this context, the knowledge mostly enables robots to understand complex scenes and extract the required information about the objects of interest. Learning typically takes RGB and/or depth information with respective labels to train neural networks. Most commonly, this knowledge is applied to determine the objects' pose in order to grasp it or manipulate it by other means such as painting, welding or inspection. Acquiring knowledge from the interactions with the environment in this context is based on deep reinforcement learning—Section 1.3. Within this strategy, the robot is highly involved in the learning process, as the knowledge is acquired through interactions with the environment based on trial and error. Therefore, giving an understanding of how to interact with the environment based on a history of events. Learning from demonstrations, however, is connected to imitation learning— Section 1.4. The basis of imitation learning is to utilize expert behaviour and make use of the demonstrated actions.

1.2. Computer-vision-based control

The recent achievements in the computer vision and deep learning fields [36, 37] have raised awareness of the benefits that such systems can provide to industrial robot systems, manufacturing and the business itself. Computer vision-based methods, in combination with the control loop of the robot, are one of the most common approaches [38] to deal with the uncertainty of the environment and to interact with it. After data acquisition, as illustrated in Fig. 1.2, the first step in the workflow of such systems is to detect objects or the presence of the objects or humans. For some scenarios, such as monitoring of safety zones in human-robot collaboration [11], it is enough to detect just the presence of an object or human to continue, pause or stop task execution. In the majority of scenarios, it is not enough, and all the required information about detected objects needs to be extracted and taken into account in the planning step. In these scenarios, the typical actions after object detection are object pose estimation, grasp pose determination, motion planning and control in accordance. Although, as illustrated in Fig. 1.2, this is only one structure of a workflow, and some solutions bypass the pose-estimation step by directly looking at possible grasping points [39].



1.2. Fig. Typical pipelines of computer-vision-based industrial robot control [18].

The novel object detection methods are usually deep-learning-based [40, 41]. Even though the main focus of object detection algorithm development historically has been addressed in different fields [42], the need for such systems is also present in the smart manufacturing context to deal with uncertainties of the environment. Thus, such deep learning-based approaches have proven to be quite flexible if conditions change [43]. Object detection in industrial robotic tasks differs from other tasks as extracted information needs to be more detailed and precise in order to manipulate objects successfully. The precision and accuracy of the computer vision system can be directly connected to the quality of the product.

Several different scenarios or combinations of conditions can be distinguished, such as working with overlapping, similar, different, multiple, complex objects with undefined positions, etc., where some situations are illustrated in Fig. 1.3. Additionally, the objects can be known, whether the information is available beforehand or otherwise, the objects are unknown. Some combinations of conditions make the task more complex and vice versa, and accordingly, available methods are similar for some scenarios. The required industrial robot control methods can briefly be distinguished in two different cases, respectively:

- Tasks that require the robots' end-of-arm tooling (EOAT) to be precisely positioned for the entire trajectory,
- Tasks that require explicit grasp planning.

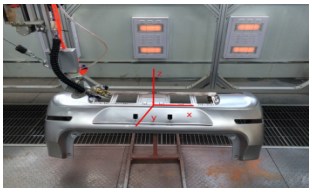
(**a**) Multiple, similar and overlapping objects [26]

(**b**) Complex, multiple, different and divided objects [44]

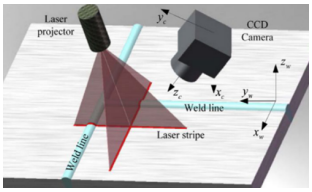
(**c**) Multiple, different and overlapping objects [45]

1.3. Fig. Some of the possible combinations of different conditions.

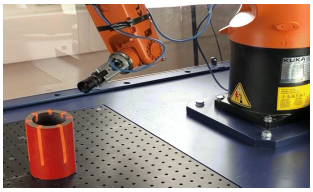
The first task can be considered to have low complexity in regards to computer vision, as the usual region of interest includes a single object that is not obstructed by other objects, as illustrated in Fig. 1.4. The application scenarios are multiple, but most applicable cases in the smart manufacturing context are connected to pose estimation of the part and more precise feature extraction for further processing, such as path generation in autonomous spray painting [46], welding [47], inspection [48], polishing, etc. Historically, in these kinds of scenarios, CAD model comparison-based methods have been proposed [49] or other solutions that are based on geometrical characteristics [50]. In some cases, CAD models of the part are not available or accurate enough; therefore, such methods are not capable of handling surfaces without mathematical representations. More recent methods can automatically generate tool paths by utilizing point cloud [51, 52].



(**a**) Spray painting [46].



(**b**) Welding [47].



(**c**) Inspection [48].

1.4. Fig. Tasks requiring precise TCP positioning along the whole trajectory.

The application scenarios for the second task have the most occurrence in modern industrial robot use cases. Typically, this can be a region of interest that includes multiple, obstructed or complex objects that can require specific grasping approaches. One of the most common robotic grasping problems in industry settings is bin-picking [53]. Accordingly, this problem has already been addressed for several decades [54]. Bin-picking is a task where arbitrarily placed objects that overlap each other in a bin must be taken out and placed in a dedicated position and orientation. Historically, this problem has been addressed as one of the greatest robotic challenges in manufacturing automation [55]. As many different combinations of conditions can be present in this problem, the proposed methods that tend to solve this task also cover most

of the robotic grasping issues.

The bin-picking problem in some cases can be eliminated in roots by avoiding situations where parts mix together as object placement in some part of the manufacturing process is structured. If object placements are structured at least along Z axes, relatively non-complex computer vision algorithms can be applied to pick and place them [56]. However, in many cases, it is not possible, as it requires modifying the manufacturing plant. Some factories lack manufacturing floor space, as complex and relatively big object dividers or shakers should be installed, and this excludes plant modification. The next aspect is high plant adaption expenses, which in some situations are not sustainable, and eventually, the plant is hardly adjustable if product design changes. Therefore, many factories end up with tasks where randomly placed objects that are located in the bin must be picked and placed by hand. This is where the vision-based grasping and automated bin-picking solution can be applied.

Vision-based grasping approaches can be divided into analytic or data-driven methods, where analytic methods focus on analyzing the shape of the target object, whereas data-driven methods are based on machine learning [57]. Learning-based approaches tend to be more generic and flexible in dealing with uncertainties of the environment and, therefore, also more promising in the smart manufacturing context. Model-based and model-free learning-based grasping approaches can be distinguished, respectively, by working with known or unknown objects.

Typically, the first step in the model-based grasping approach is object detection, which is followed by pose estimation, grasp pose determination and path planning. For model-based solutions, the optimal gripping point usually can be defined beforehand by an expert, whether it is a centre for simple shaped objects or a specific part of the object [58]. Therefore, the overall goal is to find an object that could be picked with the highest confidence and without collisions with the environment. Different approaches have been proposed throughout the years to estimate the 6D pose of the object in cluttered scenes [58–60]. Several multi-stage and hybrid approaches proceed with 2D CNN-based object detection algorithms such as SSD [61], YOLO [62], LINEMOD [59], etc., followed by an additional neural network for pose-estimation of the detected object. In addition, reverse methods are proposed by segmenting objects first and then finding the location [63, 64] and estimating the position. These methods are effective but also comparatively slow due to multi-stage architecture. Single-shot 6D pose-estimation using only RGB images [65] followed by PnP [66] shows a decrease in time and is accurate for grasping objects in the household environment; however, single-shot methods lack comprehensive industrial use-case studies.

When dealing with unknown objects, the approach and respective methods change. Grasping unknown objects requires model-free grasping approaches that are focused on finding a spot in the region of interest that could be graspable. Such methods are more appealing as they can generalize to unseen objects. Model-free methods skip the pose-estimation step and directly estimate the grasp pose. Grasps across 1,500 3D object models were analyzed in [44] to gen-

erate a dataset that contains 2.8 million point clouds, suction grasps and robustness labels. The generated dataset (Dex-Net 3.0) was used to train a Grasp Quality Convolutional Neural Network (GQ-CNN) for classifying robust suction targets in single object point clouds. The trained model was able to generalize to basic (prismatic or cylindrical), typical (more complex geometry), and adversarial (with few available suction-grasp points) with respective success rates of 98 %, 82 %, and 58 %. A more recent implementation of [45] (Dex-Net 4.0) containing 5 million synthetic depth images was used to train policies for a parallel-jaw and vacuum gripper resulting in 95 % accuracy on gripping novel objects. In [67], a Generative Grasping Convolutional Neural Network (GG-CNN) is proposed, where the quality and pose of grasps are predicted at every pixel, achieving an 83 % grasp success rate on unseen objects with adversarial geometry. Thus, these methods are well suited for closed-loop control up to 50Hz, meaning the methods can enable successful manipulation in dynamic environments where objects are moving. Multiple grasp candidates are provided simultaneously, and the highest quality candidate ensures relatively high accuracy in previously unknown environments.

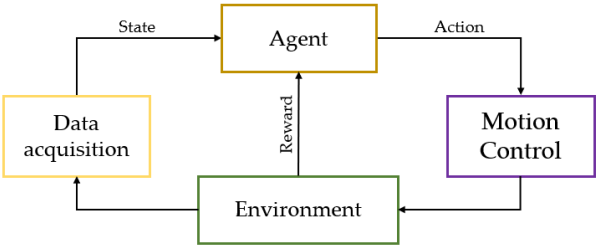
Often, the object grasping problem has been focused just on the grasping phase; however, in industrial settings, a full pick-and-place cycle is required. Post gripping [68] is as important as the gripping phase, and eventually, grasping strategies should be addressed accordingly. For model-free methods, the post-gripping process is extremely complicated and mostly cannot be executed without additional operations after picking up the object. Some exceptional cases, such as picking of potentially tangled objects, are present where proposed solutions [69, 70] only partially address this problem, mostly by avoiding the grasping of such objects.

1.3. Deep reinforcement learning-based control

After the tremendous victory of AlphaGo [71], the attention on reinforcement learning (RL) has significantly increased, whilst many potential application scenarios have emerged. Industrial robot control and the field of smart manufacturing are no exception. However, robotics as an RL domain is substantially different from most common RL benchmark problems [72]. RL methods are based on trial and error [73], where the agent learns optimal behaviour through interactions with the environment. However, the best problem representations in the field of robotics often consist of high-dimensional, continuous states and actions, which are hardly manageable with classical RL approaches [72]. RL, in combination with deep learning, defines the field of Deep-Reinforcement Learning (DRL). As DRL combines the technique of giving rewards that are based on agents' actions and the usage of neural networks for learning feature representations, it enables agents to make decisions in complex environment scenarios when input data is unstructured and high-dimensional [74].

The DRL model typically consists of an agent that takes actions in an environment, whereas the environment or "interpreter" returns an observation of how the environment changed after the

particular action was taken. Fig. 1.5 illustrates a simplified agent and environment interaction. In the context of industrial robots, the input to the agent is a current state, whereas information in this state depends on the type of action space or learning task. The state can consist of such parameters as robot joint angles, tool position, velocity, acceleration, target pose and other sensor information, which typically is environmental information acquired by a 3D camera. The action space is categorized by two main approaches: pixel-wise actions and actions that directly control robot motions. These different categories also require distinct information. For example, within pixel-wise action space, the agent selects pixels to determine where the action primitive, typically a predefined grasping trajectory, should be executed. In the context of robotic grasping and manipulation, the goal of DRL is to provide an optimal sequence of commands to succeed at the given task [74].

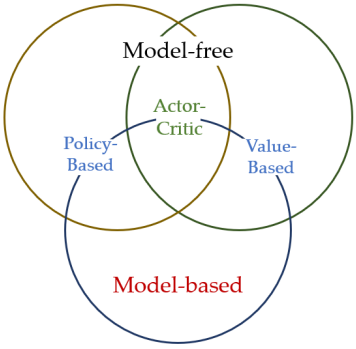


1.5. Fig. Agent environment interaction [73].

As illustrated in Fig. 1.6, the most commonly applied DRL algorithms can be grouped into model-free and model-based methods. Model-based methods have access to the environment transition dynamics, which is either given or learned from experience. Even though the model-based approaches are more sample-efficient, they are limited to the exact or approximate models of the environment. Thus, accurate models are rarely available and often are difficult to learn. Model-free methods, however, learn control policies without building or having access to a model. The ability to provide an optimal control sequence without the use of a model can be advantageous in dynamically complex environments, which is mostly the case in the context of industrial robot control. However, model-free approaches can require a large amount of learning data [75]. To overcome this issue, the benefits of combining model-based and model-free methods have been explored in [76]. Both types can be divided into two categories based on the learning approach: policy-based learning and value-based learning [77]. These methods can also be combined, which results in an actor-critic approach [78].

Even though many deep-learning-based computer vision solutions can deal with the grasping problem to some extent of effectiveness, there are limitations on how much can be achieved without interacting with the environment. When robotic grasping is approached by DRL methods, the system architecture changes as the optimal policies are learned by interactions with

the environment. Unlike computer-vision-based techniques, where robot control is mostly performed separately (e.g., by generating trajectories to successfully grasp the object of interest), in DRL-based techniques, the robots' performance is mostly evaluated as a whole.



1.6. Fig. The classification of DRL algorithms [78].

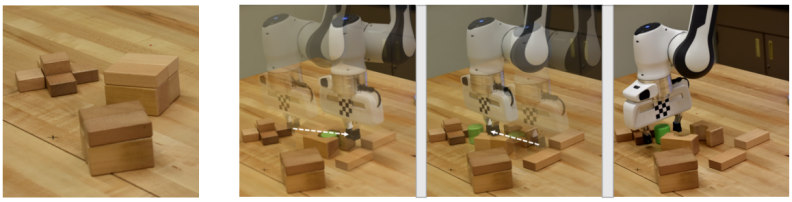
1.3.1. Typical grasping scenarios

In the last few years, DRL-based grasping approaches have been utilized in several works. Many of these are focused on solving the tasks when the region of interest includes a single [79] or a few objects with simple geometry [80] that are not obstructed by other objects. Value-based DRL methods were utilized in [81]: a combination of a double deep Q-learning framework and novel Grasp-Q-Network. The grasping performance was evaluated on three different objects: cube, sphere and cylinder, and the highest accuracy (91.1 %) was achieved by the use of a multi-view camera setup on the cube object. In a very similar scenario [82], not only the grasping problem was addressed, but also the smoothness of the trajectory by proposing a policy-based DRL method: experience-based deep deterministic policy gradient. The grasping accuracy of 90.68 % was achieved. The issue of transferring the DRL grasping agent from simulation to reality was especially addressed in [83] by utilizing an actor-critic version of the proximal policy optimization (PPO) algorithm and CycleGan to map RGB-D images from the real world to the simulated environment. The proposed method was able to "trick" the agent into believing it was still performing in the simulator when it was actually deployed in the real world and thus achieved a grasping accuracy of 83 % on both previously seen and unseen objects.

1.3.2. Push and grasp

Currently, DRL-based grasping in relatively simple environments lags behind DL-based computer vision-based methods. The most advantages of DRL-based grasping can be seen in more complex scenarios, for example, when additional actions are required before an object of

interest can be grasped. A cluttered environment can often lack collision-free grasping candidates, which can result in an unsuccessful grasp. Even though collision avoidance is usually incorporated into the grasping task by penalizing the agent if the robot collides with itself or the infrastructure, in some cases, the collisions (e.g., intended collisions) with the environment are allowed or even required to accomplish the given task. Pushing and grasping actions can singulate the object from its surroundings by removing other objects to make room for gripper fingers, therefore making it easier to perform the grasping operations. In [84], a Twin Delayed DDPG (TD3) algorithm was employed to learn a pushing policy and a positive reward was given if a graspable object was separated from the clutter. The synergy between pushing and grasping was also explored in [85], where steadier results in a more dense environment were achieved by training deep end-to-end policies. In [86], an exploration-singulation-grasping procedure was proposed to solve another relatively similar problem named "grasping the invisible". In this scenario, as depicted in Fig. 1.7, initially, the object of interest is completely obstructed by other objects, therefore, making it "invisible".

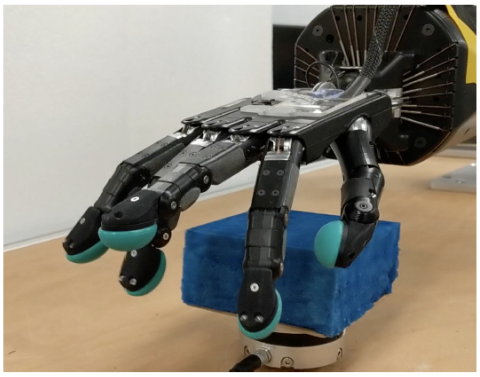


1.7. Fig. Grasping the invisible [86].

1.3.3. Force/torque information usage

Research on mimicking human sight has been conducted for a long time; therefore, the majority of DRL-based grasping methods are based on visual perception. However, one of the grasping goals is to achieve more human-like performance. Humans in the environment exploration process and especially in the grasping process also strongly rely on the sense of touch. Thus, humans can even grasp, classify and manipulate objects without visual perception. The sense of touch has been only partly mimicked by artificial sensors as humans simultaneously feel multiple parameters of the environment such as pressure, temperature, vibration, roughness, hardness and even pain [87], which is very challenging to technologically recreate. Inspired by the touch-based exploration of children, the [88] formulated a reward that encourages physical interaction and introduced contact-prioritized experience replay, which outperformed state-of-the-art methods in picking, pushing and sliding simple-shaped objects. Contact-rich manipulation can also benefit in grasping complex shape objects [89], peg-insertion tasks involving unstable objects [90] and in tasks with a clearance of and well below 0.2mm [91]. Applying different amounts of force to sturdy objects usually does not affect objects' physical properties; however, when it comes to

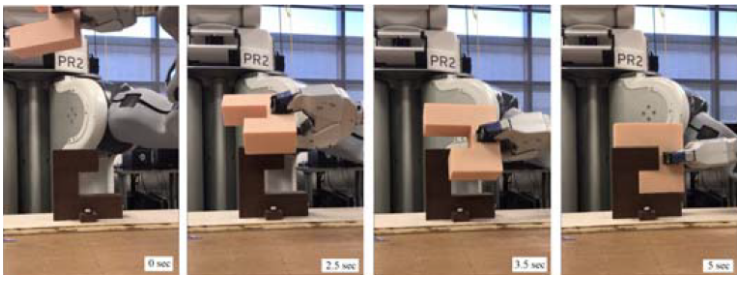
fragile objects, the robot needs to be gentle in the interaction process—Fig. 1.8. Contact-rich yet gentle exploration can be achieved by penalty-based surprise, which is based on the prediction of forceful contacts [92]. Gentle and contact-rich manipulation and in-hand manipulation especially require mimicking tactile feedback of the human hand. Usage of such anthropomorphic hands with tactile fingertips and flexible tactile skin [93] shows a 97 % increase in sample efficiency for training and a simultaneous 21 % increase in performance.



1.8. Fig. Gentle object manipulation with an anthropomorphic robotic hand [92].

1.3.4. DRL-based assembly

Similarly, as in computer-vision-based control, most of the attention in the DRL case is focused on the grasping process. However, the majority of real-world tasks require not only accurate grasping but also precise object placement. Apart from a simple stacking task that could be used in the palletization process [94], a widely used process after grasping is an assembly task. In [95], a method of learning to assemble objects specified in CAD files was proposed. Whereas the CAD data were utilized to compute a geometric motion plan used as a reference trajectory in U shape assembly—Fig. 1.9, gear assembly and peg insertion tasks. In similar settings, Reference [96] incorporated force/torque information for an assembly task of rigid bodies.



1.9. Fig. U shape object assembly [95].

DRL-based grasping approaches have been mainly utilized for grasping unstructured yet static objects, and significantly less attention has been forwarded to relatively more challenging tasks of grasping moving [97] or living objects, which can move and deform to avoid being grasped [98]. Similarly, DRL-based multi-industrial robot systems are also yet to be discovered.

1.4. Imitation-learning-based control

The ability to utilize expert behaviour and learn about tasks from demonstrations is the basis of imitation learning [99]. Opposite to reinforcement learning, where the agent learns the robot control strategies by interacting with the environment, imitation learning makes use of demonstrated trajectories, which typically are represented by states or state-action pairs [75]. The two major classes of imitation learning are behavioural cloning and inverse reinforcement learning. Behavioural cloning is often referred to as learning a policy that directly maps from the state to the control input. Alternatively, recovering the reward function from demonstrations is referred to as inverse reinforcement learning [100].

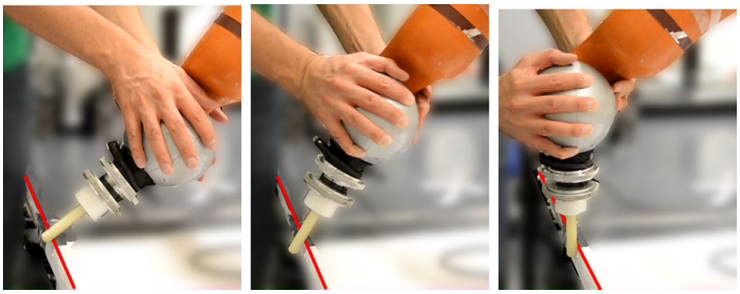
In its simplest form – known as behavioural cloning – imitation learning is reduced to a classification or regression task. Given a set of states and actions or state transitions produced by an unknown expert function, a model policy is trained to approximate this function. Even with very small parameter counts by modern standards, when combined with artificial neural networks, this method has demonstrated some success as far back as the 1980s, provided the task has simple dynamics such as keeping a motor vehicle centred on a road [101].

However, it has since become apparent that pure behavioural cloning suffers from a distribution shift – a phenomenon whereby the distribution of states visited by the policy diverges from that of the original training data set by way of incremental error, eventually leading to poor predictions by the model and irrecoverable deviation from the intended task. To improve the ability of policies to recover from this failure mode, various more complex approaches to the task have been proposed. One major direction of research has been the use of inherently robust, composable functions known as dynamic motion primitives (DMPs), employing systems of differential equations and parametric models such as Gaussian basis functions to obviate the problem of distribution shift entirely – ensuring convergence towards the goal through the explicit dynamics of the system [102]. Though showing promising results, it must be noted that the model templates used are quite domain-specific. Others have attempted to use more general-purpose models in conjunction with an interactive sampling of the expert response to compensate for distribution shift [103]. While theoretically able to guarantee convergence, the major drawback of such methods is that an expert function is required that can be queried numerous times as part of the training process. This severely restricts applicability to practical use cases since it is impossible to implement when only given a set of pre-recorded demonstrations.

A different means of handling this problem is given by the field of inverse reinforcement

learning [104]. Rather than attempting to model the expert function directly, it is assumed that the expert is acting to maximize an unknown reward function. An attempt, therefore, is made to approximate this reward in a way that explains the observed behaviour. From there, this becomes a classical reinforcement learning problem, and any training method or model architecture developed in the space of this adjacent field of study can be employed. Where early approaches made certain assumptions about the form of this reward function – such as it is linear – more recent work has proposed that generative adversarial networks be used where the discriminator can approximate the class of all possible reward functions fitting the observations over an iterative training process [105, 106].

The demonstrations can be acquired in several different direct and indirect ways, such as teleoperation, kinesthetic teaching, motion capture, virtual/augmented reality and video demonstrations [107]. Such skill transfer can not only increase usability and cope with the absence of skilled industrial robot programmers on the production floor but also increase efficiency and achieve more human-like performance. The usage of the demonstration method typically depends on the details of the task; however, kinesthetic teaching illustrated in Fig. 1.10 [108–110] can be advantageous, as demonstrations are directly acquired from robot movements, which eliminate imprecision in the form of a noise, which can be introduced by the usage of supplementary sensors. A major limitation of kinesthetic teaching is the requirement of gravity compensation, which is not supported by all of industrial robots. Another direct approach—teleoperation teaching—can be performed by different control input devices such as joystick [111], hand tracking hardware [112] and other remote control devices.

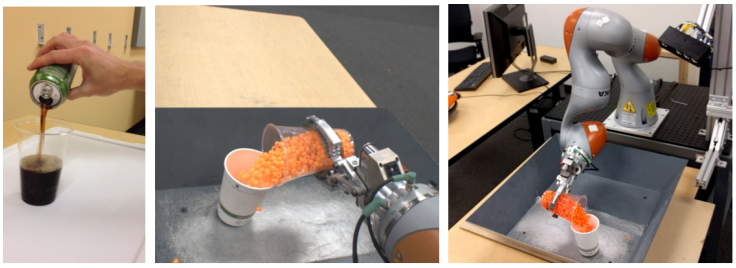


1.10. Fig. Demonstration by kinesthetic teaching [108].

In contrast to direct teaching, indirect teaching does not directly control the robot. Such systems typically learn from visual demonstrations [113] and wearable devices [114] by capturing human motions to generate trajectories for robots. However, such learning from visual representations lacks valuable information of robot interaction with the environment, therefore requiring additional extensive training. Thus, the difficulty of mimicking human behaviour based on visual information is raised due to a comparatively complex demonstration acquisition process. In this manner, the actions performed by human needs to be recognized, interpreted and mapped

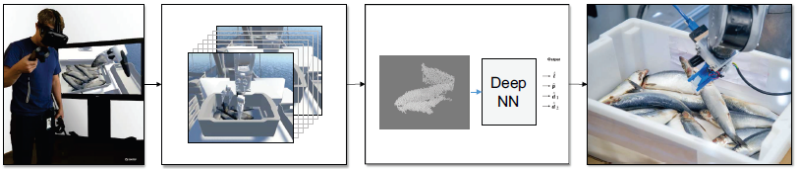
into the imitator space.

A self-supervised representation learning method based on multi-view video was proposed in [113]. The reward function for RL was created by using only raw video demonstrations for super-vision and human pose imitation without any explicit joint-level correspondence. The experiments demonstrated effective learning of a pouring task with a real robot illustrated in Fig. 1.11. The wearable device (tactile glove) used in [114] enabled observation of visually concealed changes of the workpiece. The human demonstration included both hand pose and contact forces to learn the task of opening child-safe medicine bottles. Even though indirect teaching is mostly addressed in tackling more of everyday situations that are applicable, for example, in elderly care [115], the used methods perfectly match the smart manufacturing challenges.



1.11. Fig. Pouring task with a real robot [113].

Demonstrations can also be learned in a virtual environment by training a virtual robot, as depicted in Fig. 1.12. Such virtual reality approach achieved an overall grasping success rate of 74 % grasping fish in real-world settings by acquiring 35 virtual demonstrations, which were synthetically expanded to 100,000 grasping samples [116]. Thus, with a VR-based demonstration, expert knowledge can be acquired remotely without physical presence. Complex manipulation tasks were also learned in [112] by combining virtual reality and teleoperation teaching. Such system provided an efficient way for collecting high-quality manipulation demonstrations with the highest success rate of 98.9 % achieved in the pushing task.



1.12. Fig. Imitation learning in VR [116].

When it comes to the use of motion capture data, previous work in the robotics and imitation learning corner is quite sparse, possibly due to the costly and specialized nature of the equipment involved. One previously considered direction has been the use of consumer-grade

motion tracking equipment for collecting demonstrations [117]. Unlike the study described in the Thesis in Chapter 4, they employ a single, relatively inexpensive sensor unit to generate the motion tracking data, and the main focus of the work is on accurate extraction of the demonstrations rather than modelling and extrapolation – which is left as something of an afterthought, with cluster and k-nearest interpolation methods used for inference. Others have applied the previously described DMPs to this transfer of human motion to robot configuration space [118]. Outside the imitation learning space, related work has been done in human motion modelling utilizing RNNs trained on motion capture data [119, 120].

1.5. Identified challenges and open issues

Diverse conditions on the manufacturing floor, lack of skilled personnel, functional correctness and reliability - these are only some of the challenges but definitely the core ones that are holding back the wider applicability of smart industrial robots. By looking at smart manufacturing trends and future factories, learning a new task can be neither time-consuming nor laborious. When the supply chains of smart manufacturing need to be reconfigured, the standstill is not beneficial for any organization. The pandemic situation indicated many issues and a lack of automation, which also impacted the economy as a whole [121]. However, this situation also outlined the benefits of digitization for many factories.

The ability of the smart industrial robot system to accomplish services as specified and maintain its level of performance can become a major issue within AI-based robotic systems, especially when adapting the system to new environments [27]. In manufacturing, these systems are put into continuous operation and it can be a mission-critical production environment with short maintenance times. AI must not compromise productivity through an unreliable or imprecise operation that requires regular human intervention or leads to failures. In smart industrial robot systems, this concern can be highly connected to an issue that is commonly referred to as the "reality gap" [122]. Even though reliable operation can be achieved in simulations, the trained models can perform unreliably when moved to a real environment. As real-world maintenance times can be very short, a more reliable sim-to-real transfer is required.

Even though, in some situations, robotic grasping is reliable and functions correctly, it still remains a fundamental challenge in industrial robotics. One of the most crucial aspects is the amount of the data [44, 116] required for training the respective algorithms for finding the information about the objects of interest, e.g. their position or grasp pose. Even though some approaches have proposed solutions that tend to generalize across different domains and objects, the average precision results are typically too low for precise manipulation purposes tackled by industrial robots.

1.5.1. Maintainability and usability

During the literature review an emphasis was also put forward to identify the challenges associated with the integration and deployment of smart control methods of industrial robots and thus take these aspects into account while developing novel techniques. In this context, the software quality characteristics defined in the ISO/IEC 25010 [123] standards framework were explored, with the corresponding quality measures defined in the ISO/IEC 25023 [124]. The assessment of the defined ISO/IEC 25010 characteristics in [125] and applicability of ISO/IEC 25023 measures in [126] was studied in detail and aligned with the field of the thesis. Apart from functional suitability, maintainability and usability were identified as ones the most important elements for smart industrial robot deployment and control.

The maintainability itself refers to the degree "of effectiveness and efficiency with which a product or system can be modified to improve it, correct it or adapt it to changes in the environment, and in requirements" [123], whereas for the smart control systems of industrial robots need to be designed in such a way that they can be adapted quickly to cope with different environmental aspects new customer requirements and needs, for example when new objects are introduced to the grasping system. Dedicated systems adjusted for distinct environmental conditions are no longer sustainable in smart manufacturing. Diverse conditions, especially in the context of CV-based systems, such as lighting conditions, a field of view distance and a variety of object types, can considerably decrease the precision of the system. The adaption to the changes in the environment can require modifications in software or hardware replacement. The respective component change should have no (or only minimal) impact on other components within the system, and thus, the changes should be done efficiently. During the literature review, it was identified that several sub-characteristics of maintainability, e.g. modularity, reusability, modifiability [126] are still not properly addressed in the developments of similar systems, however, are one of the prerequisites of the smart industrial robot systems.

Usability very well synergizes with maintainability however, it refers to the degree "to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [123]. The automation and deployment of industrial robot systems replace tasks that are repetitive, lack meaningfulness [127], pose a high risk of injuries [128] and also low-skill tasks that traditionally required human intelligence and dexterity. In the process of automation, more creative job openings are introduced; however, there is a lack of skilled personnel to operate and maintain advanced technologies that also include smart industrial robot systems. In fact, it is estimated that this skill gap could leave 2.4 million positions unfilled between 2018 and 2028 in the United States alone [129]. The process behind a smart robotic system can be remarkably complex, even for skilled operators. The adaptability to other specified goals should be doable without any deep specific knowledge of the underlying target technology.

Even though the proposed techniques are not quantitatively evaluated against the aspects mentioned, these both characteristics and underlying sub-characteristics are very well connected to the main aim of the thesis and thus give valuable insight into the development of smart control methods for industrial robots. Addressing particular concerns on how to increase the potential deployment of such systems and ease the usage after deployment when the system needs to be adapted. For example, purely computer-vision-based solutions in combination with industrial robots can already successfully solve a variety of tasks in a dynamic environment [130], however, most approaches still require explicit programming of robot motions and significant amounts of manually labelled data in the training process. Whereas the manual data labelling as described in Section 1.5.2 completely disagrees with the needs of maintainability and usability.

Similarly, knowledge about completing particular tasks is typically available within the company, but to use it in combination with industrial robots, an operator or outsourced experts are needed with a particular set of skills, which decreases usability and maintainability. For this aspect, the usage of expert knowledge could be applied in robotics by imitation learning through different types of demonstration methods listed in Section 1.4. Imitation learning could be an efficient way to transfer knowledge from humans to robots in a user-friendly manner, therefore tackling the issues of smart robot deployment as well. However, such approaches have been poorly explored in the industrial robot control context.

1.5.2. Availability of the data

The collection and processing of data for machine-learning tasks play an important role, and smart industrial robot control is not an exception, as ML-based approaches are widely used to solve challenging tasks. According to [131], on average, more than 80 % of time spent on AI projects is based on the collection and processing of the data. The data collection techniques can be distinguished using several methods. The reviewed techniques in [132] vary for different use cases. For example, in the field of autonomous driving [133] or in more traditional applications, such as household object recognition, people detection or machine translation, there are openly available datasets that could be reused and adapted for specific needs for model training [134].

The use of simulations and synthetic data can facilitate the development of smart industrial robotic systems in various ways and stages. Generating low-cost and large amounts of data, accelerating the design cycle whilst reducing costs, and providing a safe and fully controlled testing environment are only some of the opportunities [135]. However, these benefits come with several challenges that are yet to be solved. In the context of the reviewed smart industrial robot control methods, the challenges are highly connected to differences between simulations and real life, which, therefore, usually decreases the precision of the system when deployed in real-life scenarios. Physic simulations, object virtual representations, sensor data recreation, artificial lightning and other countless bits of real environment all contribute to the issue of

transferring learned models from simulation to reality.

Synthetic data for computer-vision-based robot control typically has been composed by placing foreground objects on background scenes with different parameters that can be varied. Some approaches proceed with 2D images that are placed on a set of background images [23] [136], with a set of rules or physical simulations for piling the objects realistically. The level of complexity for such and similar methods is lower, however, they are typically restricted to 2D nature, which contributes to the lack of realism, which therefore decreases detectors' performance. 3D graphical engines have also been utilized in synthetic data generation, however, these are typically dedicated to household situations or usage with everyday objects [137] [138].

Due to specific elements of smart manufacturing and precision requirements, a different situation can be seen in this field. The product variety is changing more quickly, and algorithms have to be trained on new data sets. Therefore, the re-usability of existing data sets is fairly low. Manual labelling methods cannot meet the requirements of agile production as it is time-consuming, expensive and usually requires expert knowledge of the specific field, and thus, human error can arise in the labelling process [139]. These aspects also conflict with the goal of smart industrial robots to automate tedious jobs that lack creativity. Introducing the smart industrial robot control methods that require manual labelling shifts the manual work to a different sector. Synthetic data generation, on the other hand, decreases the time required for the labelling process by the use of physics and graphics engines to replicate real-life situations of environmental uncertainties. In a controlled environment and in non-complex scenes listed in Section 1.2, the benefits of synthetic data usage might not overcome the reality-gap issues. However, generalizing the knowledge requires vast amounts of data, and when it comes to many different environmental parameters and complex scenes, all the variables can be taken into account in the generation process. Thus, combining real and synthetic data could leverage the efficiency of the data collection process and the precision of the trained model, but yet lacks comprehensive studies in manufacturing applications.

Moreover, in scenarios where robots learn by themselves (see Section 1.3), there is also a possibility of training the robots in real life whilst interacting with the real environment and using the feedback from it to learn policies. Similarly, this can be tedious, expensive and very time-consuming work [140]. Additionally, in this case, the robot needs to actually execute actions that, in the first learning iterations, can be dangerous to the environment and the robot itself while it is exploring the action possibilities. To decrease the risks that can arise from random exploratory actions, the training of the robot is mostly moved to digital space. However, complex and unstructured environments, uncertain external loads, actuator noise, friction, wear and tear, impacts, etc., are some of the many sources [135] that are not yet accurately represented by simulations. This means that the solutions learned in a simulated environment can perform differently and usually with less accuracy in real-world settings, if performing at all.

2. SYNTHETIC DATA GENERATION FOR DEEP-LEARNING-BASED TASKS

Data collection and processing play an important role in machine-learning tasks, including smart robot control, as nowadays ML-based methods are widely used to solve challenging tasks with the use of robots. According to [131], on average, more than 80 % of time spent on AI projects is based on the collection and processing of the data. For robots, the data is an important element as it enables them to anticipate events and prepare for them in advance, therefore, hoping with harsh, dynamic conditions and unforeseen situations. In essence, the data supports the perception part that includes acquiring the information of an unstructured environment, analyzing this information, extracting features, interpretation, etc. This information further on is taken into account in adaptive path planning and motion control steps to cope with highly unstructured environments.

On the one hand, coping with unforeseen situations requires special algorithms that are able to generalise over a variety of scenarios. On the other hand, data is the key factor, and the success rate depends on the fact that the robot has had a similar experience in the training process. The training set, however, typically consists of annotated data sets. Manual labelling of such data sets is time-consuming manual labour and thus, some corner cases might be missed out from the data-set, due to a complicated training data acquisition process. Data synthesis, however, alleviates this process and simplifies the use of modern computer vision methods in the industry.

Image synthesis or rendering is a process of generating digital images from virtual scenes. The photo-realism of rendered images, videos, and computer games keeps increasing. Also, the tools for creating virtual environments with included physics simulation are becoming more user-friendly and affordable. For example, such tools as Blender, Unity, and Unreal Engine can be used free of charge. Therefore, AI and computer vision research communities increasingly use these tools to generate data and train systems in virtual environments directly, however, the precision decrease can be observed when models are trained on purely synthetic data when compared to real data [141]. In smart robotic systems, this concern can be highly connected to an issue that is commonly referred to as the "reality gap" [122]. Even though reliable operation can be achieved in simulations, the trained models can perform unreliably when moved to a real environment. As real-world maintenance times can be very short, a more reliable sim-to-real transfer or more realistic simulations are required.

In this chapter, different synthetic data generation approaches for further usage in computer-vision-based robot control are explored. Namely synthetic data generation of simple-shaped-rigid objects and transparent, deformable objects. Further on, this data is used to train object detection models and compare them with object detection models trained on manually labelled data. Thus, it is also analysed how the real data, synthetic data and different combinations of both affect the precision of object detection models in bin-picking setup.

2.1. Synthetic data generation and detection of simple-shaped rigid objects

Even though modern object detection methods are effective at detecting traffic signs, pedestrians, household objects, etc., the methods fall short in industrial settings due to a lack of training data. The proposed method utilizes modern graphics engines to generate realistic data for training neural networks for robotic grasping in industrial-specific scenarios, where data can be use-case-specific and change over time. The focus is concentrated on generating versatile data with a vast amount of adjustable parameters. Thus, the parameters can be adjusted by specific needs. In this case, the data of unobstructed objects in the corresponding orientation is extracted to train AI models only on the objects that have the highest probability of resulting in a successful grasp when the model is deployed. The object detection described in the experiments of this chapter serves as the first step in the whole pick and place process, and only information about the most promising objects that could be grasped is then processed further.

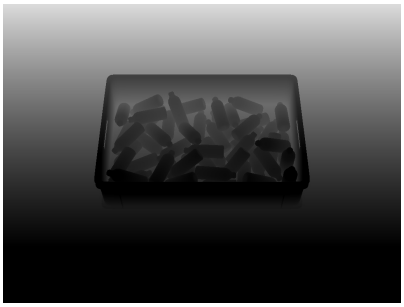
The aim of this task is to automate the systematic rendering of highly realistic synthetic pictures to generate data sets for training the object detection algorithm. The image generator obtains images by arranging any kind of objects that have 3D models within a virtual 3D scene from which it renders highly realistic images. In this case, it is a box with simple-shaped objects - white bottles. By tuning parameters of the 3D scene such as an object, camera and light positions, object colour or texture and surface properties, brightness, contrast, and saturation, a large image diversity can be generated in resolution and levels of realism depending on the user's needs. By further exporting relevant ground truth data from the 3D scene including the location and orientation of objects within a rendered image, the generated data is fully annotated.

2.1.1. Synthetic generation of realistic pile images and annotations

By sampling all possible configurations of objects and image parameters within the 3D scene (in arbitrary, user-defined granularity), a modification space is defined, allowing for the automated generation of large synthetic data sets, for which the diversity of images is controlled by the user. By systematically defining appropriate scenes and modification spaces, the image generator can be used to generate not only training and validation data sets in sufficient quantity (overcoming a lack of training data, which is often a limiting problem) but in general, also allows for generating data sets specifically designed for specific experiments later on (e.g. to compare across different classification models or to extract specific insights of the algorithm.

In this experiment, the focus is on automating the generation of realistic data sets for training the object detection algorithm, i.e. highly realistic images of piled objects together with their corresponding depth map, for which every single object visible in the scene is annotated with location and orientation ground truth. Corresponding instance-segmented images are also gen-

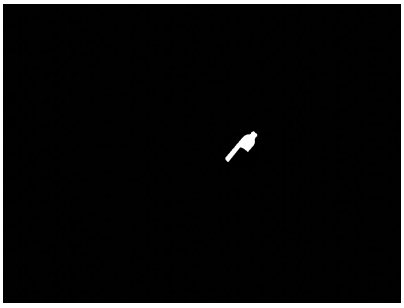
(**a**) Generated image



(**b**) Generated depthmap

2.1. Fig. Synthetic data of randomly piled, similar objects [24].

(**a**) Ground truth bounding box



(**b**) Ground truth segmentation image

2.2. Fig. Data for learning-based computer vision algorithms [24].

erated as depicted in Fig. 2.2 b. For image generation, a free open-source tool *Blender* is used, for which the rendering of parametrizable 3D scenes is fully automated in *python* by making use of Blender's Python API.

A set of functions in Python was implemented for setting up virtual 3D scenes and controlling the image rendering process, where the configuration of objects and image parameters for the defined configuration space were kept as variables. These functions have been implemented in such a way, that the whole process of image generation iterates across all possible configurations of varying image parameters within the given configuration space (i.e. the placement of the 3D objects into different positions of the 3D scene, applying different textures and shaders to the objects, different positioning of light sources and the camera, running the rendering process) is being automated and executed in headless mode and no manual interaction via Blender's GUI is required.

To additionally allow for a more realistic object placement within the considered 3D scenes, especially for piling up any number of objects more realistically and naturally, Blender's integrated physics engine was used, which enables to optionally apply simple rigid body simulations based on convex hull collision detection. For post-processing and annotation of images, OpenCV

was used.

2.3. Fig. Different perspectives and light conditions.

2.1.2. Datasets

The real data was gathered by randomly distributing bottles in a box. For each of the acquired images, the positions of the bottles were altered. The intensity of lightning and camera exposure time was systematically modified to acquire a high diversity of different lighting conditions. In total for training purposes, 2200 real images were acquired and manually labelled from which 1760 images were for training and 440 images for validation purposes. Furthermore, the training and validation datasets were rotated four times by 90 degrees, in total resulting in 7040 training images and 1760 validation images. Two test datasets were gathered and manually labelled, first dataset *Test 1* was captured in similar conditions as the real training dataset, however, *Test 2* was captured with a different camera and in different conditions.

The synthetic dataset was generated in the same amount as the real dataset, consisting of 8800 photo-realistic high-resolution scenes. For every individual scene, an initially empty box was filled with randomly placed bottles by making use of Blender's physics simulation engine to achieve realistic positioning and orientation. Realistic textures and Blender's principled BSDF shader nodes are used to achieve realistic renderings of the scenes including lighting, reflections and indirect light bounces.

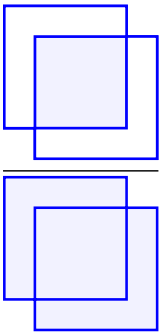
After filling the box with the bottles, a series of renderings are created for which power levels of 4 different light sources and the orientation of the camera are varied, which orbits around the box and renders the scene from 16 different angles as depicted in Fig. 2.3. For every camera orientation, also a depth image and the segmentation images of the individual bottles are

generated as seen by the camera and labelled by the object ID as illustrated in Fig. 2.2. A separate annotation file is generated for every camera perspective which contains the individual object's orientation and rotation in-camera coordinates together with the object's visibility percentage.

2.1.3. Evaluation metrics

The most commonly used metric to measure the accuracy of object detection in the images is average precision (AP), which is also utilized in this section to evaluate the performance of object detection in the experiments. In this case, variously set (0.5 - 0.95) intersection over union (IoU) thresholds are used to perform AP measurements.

$$IoU = \frac{area\ of\ overlap}{area\ of\ union} = \tag{2.1}$$



$$class(IoU) = \begin{cases} Positive \rightarrow & IoU \geq Threshold \\ Negative \rightarrow & IoU < Threshold \end{cases} \tag{2.2}$$

With the IoU the overlapping area is measured between the ground truth and the predicted bounding box and the precision is evaluated over different thresholds. True positive (TP), false positive (FP), and false negative (FN) estimates for each detected object are used to calculate precision and recall to perform AP measurements.

$$precision = \frac{TP}{TP + FP} \tag{2.3}$$

$$recall = \frac{TP}{TP + FN} \tag{2.4}$$

$$AP = \int_0^1 p(r)dr \tag{2.5}$$

where $p(r)$ is the interpolated precision at each recall level r. The integral represents the area under the precision-recall curve.

2.1.4. Results

The performance and precision of the proposed synthetic data generation approach were evaluated by various aspects. First, the object detection performance was investigated with deep learning models trained using different ratios of synthetic and real data combinations and by utilizing the maximum amounts of the acquired data. Starting with 100 % of real data, the real data ratio was incrementally decreased by 10 % at the time by substituting it with the synthetic data as depicted in Table 2.1.

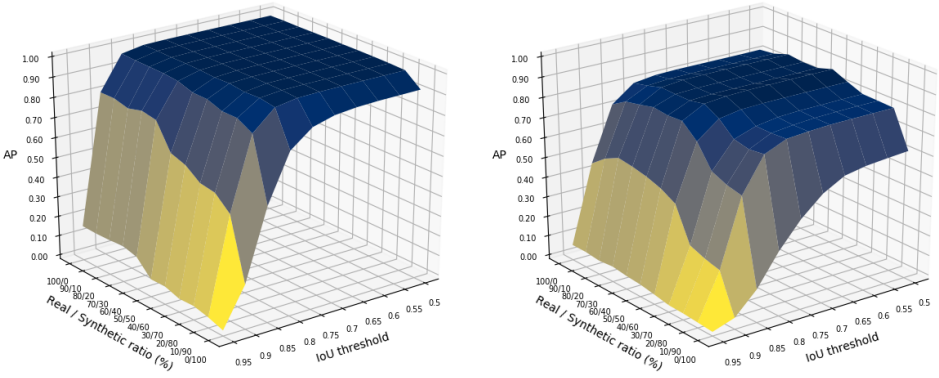
2.1. table

Evaluation of object detectors precision [24]

Data distribution			***Test 1***			***Test 2***		
Real	**Synthetic**	**Real / Synthetic Ratio %**	**Step**	**AP @0.5:0.95**	**OD %**	**Step**	**AP @0.5:0.95**	**OD %**
8800	0	100 / 0	9100	88.61	100.00	9100	69.22	96.95
7920	880	90 / 10	7900	88.61	100.00	9200	71.04	98.47
7040	1760	80 / 20	6000	88.36	100.00	8000	73.23	100.00
6160	2640	70 / 30	6300	**88.65**	100.00	7600	72.83	100.00
5280	3520	60 / 40	6900	88.22	100.00	7400	72.5	100.00
4400	4400	50 / 50	4400	85.82	100.00	5000	**73.84**	100.00
3520	5280	40 / 60	8000	85.59	100.00	8700	70.27	100.00
2640	6160	30 / 70	7200	84.39	100.00	5900	64.57	100.00
1760	7040	20 / 80	7200	84.23	100.00	4500	63.62	100.00
880	7920	10 / 90	8200	82.59	100.00	7000	63.25	100.00
0	8800	0 /100	7700	70.01	100.00	5000	38.66	100.00

The evaluation was performed on the described datasets - *Test 1* and *Test 2*. Object detector evaluated on data close to real training data *(Test 1)* scored similar average precision results when real data amount was higher than synthetic data as depicted in Fig. 2.4(**a**). This also holds true for higher IoU threshold values from 0.85 to 0.95. All of the trained models showed similar average precision results in the IoU threshold region from 0.5-0.8. The main difference can be seen in the case when the model is trained on purely synthetic data, as the precision remarkably decreases.

A different situation can be seen when the object detector is evaluated on a test dataset that contains different environmental parameters - *Test 2*. In this case the synthetic data supplements real data and increases average precision, whilst achieving the highest precision on a 50/50 ratio. Similarly as with the evaluation results on *Test 1* dataset, also in this case object detector trained on purely synthetic datasets showed the least precision.



(**a**) Test 1 (**b**) Test 2

2.4. Fig. Average precision of object detection models on real images over different IoU thresholds, viewed by the ratio of real to synthetic data in the training datasets [24].

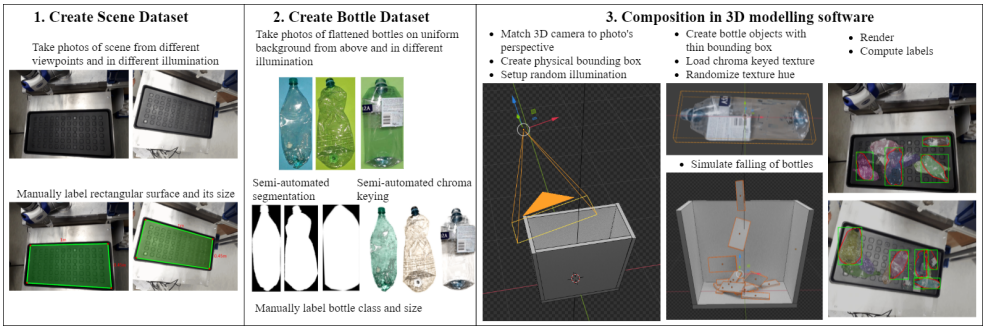
(**a**) Test 1 (**b**) Test 2

2.5. Fig. Results visualized over different data sets.

Even though the object detector trained on real data or different combinations outperforms the detector trained on purely synthetic data, the main precision aspects in the Thesis are connected to the information that can be further used for robot control. Respectively, the goal is to detect at least one object in the scene with an IoU threshold above 0.95. Obtained results on this aspect are depicted in Table 2.1 under columns Object Detected (OD). On both test datasets, the trained models could meet this requirement, except in the Test 2 case, when the model was trained on purely real data and in the following 90 / 10 ratio.

2.2. Synthetic data generation and detection of deformable and transparent objects

The synthetic data generation methodology was also evaluated with deformable and transparent objects, however, the generation steps were slightly different due to the nature of such objects, mainly in relation to deformations that can be present in the objects. The deformation itself is not yet feasible to implement in the generator itself, as the recreation of a such complex process at the moment is not directly supported and would be a comparatively very computationally expensive task. Virtual reconstruction of the environment is made from 2D photos of the scene containing a rectangular planar surface. Due to notable difficulty in 3D scanning transparent objects, bottles are reconstructed as planar objects with a 2D texture of real bottles. This solution does not harm the intended application as plastic bottles are expected to be crushed as well as camera can be placed above the scene. The proposed pipeline is described in detail in [21] and an overview is illustrated in Fig. 2.6.



2.6. Fig. Proposed approach of generating synthetic data for transparent bottle detection in piles [21].

2.2.1. Synthetic dataset creation

The dataset includes both background images that are taken from above with slight variations in perspective and illumination and foreground images of flattened transparent bottles. The background images include labels for the dimensions and corner points of the rectangular surface in the image, and the foreground images are segmented and labelled for size and class. To compose the final dataset, a Blender-Python script is used to clutter the background scenes with bottles and introduce variations such as bottle colour, size, and scene illumination. The resulting dataset is used for realistic virtual scene reconstruction and object detection experiments.

Several synthetic datasets were generated with various sources of variation consisting of 5,000 images each. Unless stated otherwise, synthetic data with all sources of variation enabled

2.7. Fig. Samples from different datasets. For Synthetic set samples generated using all sources of variation are shown [21].

was used. 80 % was used for training and the rest for validation. Real data was collected in the same environment as the background images. The tray on the table was cluttered with the same bottles used in synthetic dataset generation, and photos were taken with the camera mounted above. Bounding boxes and classes were manually labelled for each scene. The dataset consists of 2031 photos taken in bright illumination from a single viewpoint and with a small number of unique clutter scenes. The real dataset was split in the same manner as the synthetic dataset. The test set consists of real data that captures all the variation expected from a good object detector to cover, in other words, it represents the target domain. It consists of 51 uniquely cluttered scenes with 3 illumination conditions (bright lamp, dim lamp, ambient light) and 4 camera positions from above, 612 images in total. Samples from synthetic, real and test datasets are shown in Fig. 2.7.

2.2.2. Training and evaluation metrics

All models in this experiment are trained using default hyper-parameters and configuration in the "YOLOv5" implementation. For evaluation on the test set, only the best weights among all epochs were used. During the experiments, multiple object detector model variants are trained. Similarly, as for rigid objects average precision is measured with mAP@0.5:0.95 metric on the test set. mAP is computed as the mean of AP scores per object class.

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \tag{2.6}$$

where n is the total number of object classes, and AP_k is the average precision for the k-th object class. AP is summed for all object classes and then divided by n to obtain the average of the AP values.

Models trained on either synthetic or real datasets are further referred to as single dataset models or Synthetic and Real models respectively. Models trained on both synthetic and real datasets are referred to as either Concatenated or Combined models depending on the method.

The simplest way of combining two datasets is to merge i.e. concatenate train sets and validations sets, respectively. Two concatenated model variants were trained, first, that validates on concatenated validation sets from both datasets and the second one, that only validates on the real validation sets, but due to similar performance and for a fair comparison with combined models (explained in next paragraph), only latter model is reported.

Synthetic data is usually more abundant than real data, thus by simply merging both datasets, a "synthetic" gradient can suppress the "real" gradient during optimization. For this matter, the multi-task-inspired loss function is used to balance the contribution of "real" and "synthetic" gradients. More specifically, "YOLOv5" mini-batch loss $l(B)$ is scaled using following expression:

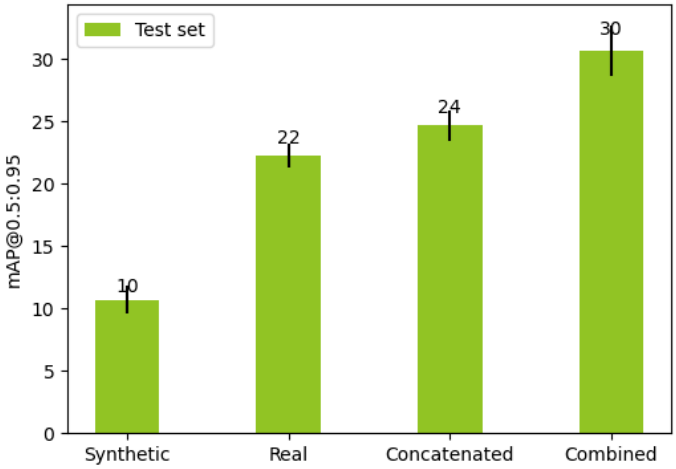
$$\ell'(B) = \begin{cases} \frac{|D^S|+|D^R|}{2|D^R|}\ell(B) \cdot \alpha, & \text{if } B \subseteq D^R \\ \frac{|D^S|+|D^R|}{2|D^S|}\ell(B) \cdot (1-\alpha), & \text{if } B \subseteq D^S \end{cases} \tag{2.7}$$

, where D^S and D^R denote synthetic and real training sets respectively, B is a mini-batch sampled from either one set, and $\alpha \in [0, 1]$ is an additional scaling parameter, that weights contribution of each set to the gradient. In experiments, $\alpha = 0.5$ is used for equal contribution, unless stated otherwise. A bigger α value means a larger real set contribution, while a smaller value means a larger synthetic set contribution. Best weights are saved by validating only on a real validation set as it facilitates the goal of generalizing the detector to a real domain.

2.2.3. Results

Multiple experiments are performed to evaluate the generated data against data collected in a conventional way. First, the performance of models trained on real, synthetic and combined datasets is compared, where Fig. 2.8 shows the results of this experiment.

Results show that models that use a combination of synthetic and real data perform better on average. Combined models on the test set achieve 30 mAP@0.5:0.95 on average, which is 1.36 times more than Real models and 1.25 times more than Concatenated.



2.8. Fig. Average performance of models. Lines represent one standard deviation [21].

Comparison of dataset sizes

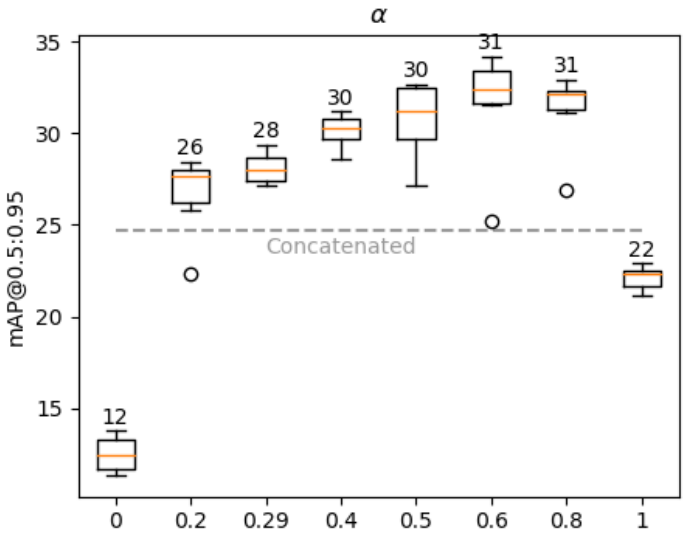
In order to find out whether the Combined model's performance could be traded away in favour of a smaller real dataset size, experiments are performed with different sizes. The real dataset is divided into 3 subsets, namely 25 %, 50 %, 75 %, by taking only the corresponding percentage of data from the beginning of the full dataset for both training and validation sets. Additionally, real models are trained on those subsets to confirm that reducing data reduces performance, which might not be the case if images contained a lot of redundant information due to overlap. For Combined models, the full synthetic dataset is used. Table 2.2 summarizes the achieved results.

Best results are achieved by Combined models trained on 100 % of real data. Combined models with only 25 % of real data achieve similar performance to Real models on the Test set, which suggests that in this case amount of real data can be reduced by a factor of 4 if trained in combination with synthetic data. However, for the validation set only 25 % of the Real data can be removed without performance loss.

2.2. table

Performance(mAP@0.5:0.95) with respect to real dataset size [21]

Dataset	Test set (Avg±Std)
Real 25 %	13.6±2.0
Real 50 %	16.8±1.5
Real 75 %	18.8±0.9
Real 100 %	22.3±1.0
Synthetic + Real 25 %	22.4±1.0
Synthetic + Real 50 %	23.0±3.5
Synthetic + Real 75 %	28.6±2.4
Synthetic + Real 100 %	**30.7±2.0**



2.9. Fig. Box plot of Combined models with respect to alpha values in the combined loss. The average performance is shown above the boxes. The Grey dashed line represents the average performance of Concatenated models [21].

Contribution of datasets

Additional tests are performed to find an optimal contribution ratio between synthetic and real training sets. More specifically, Combined models are trained with multiple α values in the combined loss (eq.2.7). A bigger α value means more real set contribution, while a smaller value means more synthetic set contribution. For models with $\alpha = 0.5$ combined models from

previous experiments are reused, while $\alpha = 1$ and $\alpha = 0$ models, corresponding to Real and Synthetic models respectively, are trained from scratch due to minor differences from using Combined loss. Results are summarized in Fig. 2.9.

Models with α ranging from 0.4 to 0.8 achieve good results with minor differences, however, favouring real data slightly more ($\alpha = 0.6$) gives the best result. By favouring synthetic data 4 times more ($\alpha = 0.2$), worse results were obtained than by favouring real data by the same amount ($\alpha = 0.8$), confirming the real data is more important. As expected, $\alpha \in \{0, 1\}$ models score similar performance to their regular loss function analogues (see Fig. 2.8), however, both are outperformed by $\alpha \in \{0.2, 0.8\}$ models, suggesting that even small contribution from opposite set significantly improves performance.

By simply merging both datasets, performance should be close to $\alpha = 0.29$ models due to having a $1624/5624 \approx 0.29$ proportion of real training data among the whole concatenated training set. However, surprisingly, worse performance for concatenated models was observed, which might be caused by different loss scales due to Combined loss. Nevertheless, even $\alpha = 0.29$ models perform worse than by favouring real data, confirming that balancing the contribution of datasets is useful when using larger synthetic datasets.

2.3. Summary

In dynamic environments, especially in the case of randomly piled objects, a lot of uncertainties and different environmental conditions can be present. Ideally, these different conditions should be covered by the training data set in a deep learning-based object detection task to satisfy the precision requirements. However, gathering and labelling the real data is a tedious task and requires a certain amount of human resources, and in some cases, it is complicated to recreate all the possible configurations. With the synthetic data generation approach, it is intended to mimic the real data characteristics and diversify the dataset by a systematic rendering of highly realistic synthetic pictures. By tuning image parameters such as an object, camera and light positions, object colour or texture and surface properties, brightness, contrast, and saturation, a large image diversity can be generated in resolution and level of realism depending on the requirements.

For simple-shaped-rigid objects, the generated dataset, real dataset and different combinations of both were used to train an object detector. The trained models were evaluated on two different test datasets. In most cases when the real data ratio was higher than synthetic data, the model achieved higher precision ratings. The real data still outperforms synthetic data if we are solely analysing the object detectors' average precision, however in smart control of industrial robots, it is not that crucial to have precise information about all the objects in the scene, as typically only one object can be grasped anyway, after which the scene has changed and needs to be perceived again. Whereas the achieved precision is sufficient in the case of simple-shaped-rigid objects. Thus, by diversifying the training dataset with synthetic images a precision increase can

be observed, however, when the synthetic ratio is over 50 %, the precision decreases.

Similarly also for deformable and transparent bottles synthetic data generation method was proven to be useful in the process of training object detectors, however, in this case, models trained on purely synthetic data, achieved precision results that were notably worse than in the case of rigid objects. Nevertheless, synthetic data generation provides additional information due to the vast amount of variations.

Overall, the developed synthetic data generation frameworks show promising results in deep learning-based object detection tasks and can supplement real data when the variety of real training data is insufficient. However, adding the synthetic data to real data requires testing to find the peak of precision, as adding more synthetic images results in lower precision.

3. COMPUTER-VISION-BASED CONTROL

Programming robots is challenging as it takes time, requires expertise in both the task and the platform, and can be prone to errors. Industrial robots use various vendor-specific programming languages and tools that need specific proficiency. To broaden the use of robots and increase the level of autonomy in industrial settings, it is also essential to make it easy for non-experts to instruct robots and interact with them. This also motivates the development of smart industrial robots and the integration of AI techniques in robotics, making systems as autonomous as possible, but only as long as it is reasonable [142].

Computer-vision-based industrial robot control is one of the approaches to increase the applicability of industrial robots by reducing the complexity of the system when deployed. Indubitably, the complexity of developing such systems can increase remarkably based on the flexibility required. In comparison to traditional industrial robot control, an additional processing system is required for perceiving and understanding the data coming from the physical field. Computer-vision-based control can be divided into several categories corresponding to different cases and conditions as described in Section 1.2. The main goal of this chapter is to enable robots to *see* and process the environmental information, therefore, enabling them to perform tasks in dynamic conditions.

Performing tasks in dynamic conditions, in the sense of robotic movements, isn't straightforward as well. For example, for traditional industrial robot control, exact and precise knowledge about the position of the object is required for the motions even in a simple pick-and-place process. The motions are typically programmed in a non-flexible manner and thus unable to deal with even small environmental variations. Considering the randomness of dynamic environments, for instance, as illustrated in Fig.1.3(**a**), the robotic arm can get into collisions while travelling to the goal point, or even the goal point can be in a collision between the robot and environmental elements such as the container where the objects of interest are positioned. It is nearly impossible to define all the possible robot motions and even more cumbersome to validate that the robot won't collide during the task executions. For this case, this chapter also emphasizes the motion planning aspects of computer-vision-based robot control. Overall, the experiments and robotic systems described in this chapter are driven by synthetic data described in Chapter 2 and are further built upon in the development of computer-vision-based robotic systems.

3.1. Grasp-pose estimation and grasp planning

Computer vision algorithms give valuable information about the environment and particularly, in this case, about the objects and where they are located in the workspace. The object detection algorithms have been trained to detect only the objects that are most likely to result in a successful grasp. Where also in this matter, the results show that in every scene with a model trained on completely synthetically generated data, at least one object with an IoU threshold over

0.95 is detected.

In some scenarios after the object detection phase, it is possible to proceed to estimate the grasp pose of the object of interest. However, with this approach, several properties of the object wouldn't be determined, therefore limiting it to distinct environmental parameters, specific use cases and vacuum gripping. Currently, the object detection algorithms can work in real-time and can process as high as 90 frames per second [143]. Therefore, proceeding with only object detection can be beneficial to ensure the fulfilment of low cycle time requirements, however, with limited action possibilities.

With object detection as described with the synthetic data generation experiments in Chapter 2, the object typically is localized in a 2D image, and the type of object is given. The information is sufficient to find the centre position of the object, however, to determine other properties of the object, such as the orientation of the object, additional image processing algorithms are required. If simple-shaped objects are positioned in one plane, and the grasp point is located in the centre of the object, the object detection result can be directly used to compute the grasp pose. Additional systems on top of the object detector for ensuring that any constraints haven't been violated or constructing the candidate system to grasp the objects in the sequence specific to the respective use-case, however, should be incorporated. The grasping of such objects typically can be done by standard robot control methods and standard RGB cameras, therefore are not further investigated in the Thesis.

In case when objects overlap each other in a pile, as illustrated in Fig. 2.5, the position of objects varies in three dimensions, respectively, the objects are positioned also in different depth levels. For scenarios when the object just needs to be grasped and placed without any additional manipulation steps or a need for precise placing, the pick-and-place cycle can be completed with object detection as the main computer-vision functionality. After object detection, the grasp pose is estimated by aligning the RGB information with respective depth information in the region typically acquired by RGB-D sensors/3D cameras. Depth information in RGB-D images is aligned with the corresponding image pixels, providing information on how far from the camera the region of interest is located. The region of interest, in this case, is a circle-shaped area around the centre point of the detected object. From this area, the approach angle is calculated by principal component analysis (PCA).

For precise manipulation of object or placement, the 2D object detection part itself falls short, as, in this way, the object is localized only in 2D space with a bounding box that includes the object and some area around it. The detection results cannot be directly used to determine the orientation of the object, which is required for precise placement or other manipulation procedures. Therefore, the detected area is utilized further by the object segmentation algorithm. Segmentation determines which class every pixel of an image belongs to. Instance segmentation, however, is a type of segmentation that differentiates among pixels belonging to different instances of the same class. With this information, a visible shape of a specific object is acquired

and used to determine the grasp pose of an object, which in turn is handy for picking up and manipulating the object. For the object segmentation task, the Mask R-CNN algorithm is utilized, where the neural network is trained on purely synthetically generated data.

(**a**) Bottles (**b**) Cans and bottles

3.1. Fig. Different scenarios that include segmented objects with calculated grasp positions.

The information returned by object segmentation allows to precisely determine the longitudinal direction of the object and other specific features of the object. For example, in the case of the bottle, also the top and bottom parts of it can be identified. After mapping the acquired information to corresponding depth information, the grasp pose is calculated, similarly as described previously. The grasp pose is also illustrated in Fig. 3.1 with the axis symbol.

3.2. Robot-control

Whether the grasping of objects is achieved by human resources or robotic systems, it is one of the main requirements in many manufacturing processes. However, in highly dense scenes, where the environment includes several uncertainties, such as randomly distributed objects, the robot navigation is closely related to the perception of the scene. This can be affected by features of the objects of interest as well as the previous and following manipulations. Besides the picking and placing of the object, the robotic system must perceive the environment and extract the required features from it to interact with it. Precise information about the location of the object of interest in space is one of the prerequisites for obtaining a proper grasp position and therefore accomplishing a successful grasp - described in the previous section.

The grasping process itself is usually split into several parts that typically depend on the types of gripper and object of interest. In the Thesis, the robot control is mostly connected to two grasping techniques, namely, vacuum grasping and 2-finger grasping. There are some minor differences between both, however, the main grasping phases are similar: approaching the object, coming into contact with the object, increasing the force/vacuum until the object is securely grasped and moving the object until it can be released in the place position. In

bin-picking settings, a surrounding-aware visual perception system, in combination with the correct approach and retreat movements, is a remarkably important part of the navigation system. Objects are randomly distributed, and unintended collisions that can occur with the bin, other environmental elements, and the robot itself should be strictly avoided.

Firstly, the data acquisition device, in this case, a 3D camera, and the robot or robots must be operating in the same coordinate system, for which the hand-eye calibration must be performed, described in Subsection 3.2.1. Secondly, the robot control framework needs to be chosen. The way how industrial robots are programmed strongly depends on the robot manufacturer. As we can see in table 3.1, every industrial robot manufacturer has developed its own programming language, simulation software and possibilities that their software offers differ from brand to brand. In terms of flexibility of the system, hardware abstraction is important so industrial robots from different manufacturers can be programmed and controlled in one way. There are multiple frameworks that offer compatibility with different industrial robot brands, but the most popular of them is ROS-Industrial [144], which is a part of the Robot Operating System (ROS).

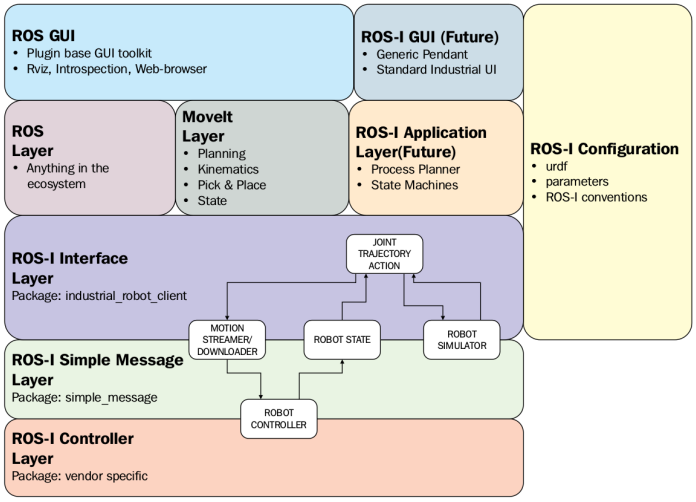
3.1. table

Industrial robot manufacturers and their software [17]

Manufacturer	**Programming language**	**Simulation software**
Universal Robots	URScript	URSim
ABB	RAPID	RobotStudio
Kuka	KRL	KUKA Sim
Kawasaki	AS	K-ROSET
Stäubli	VAL3	Stäubli Robotics Suite
Comau	PDL2	Process Simulate
Yaskawa	Inform	MotoSim
Fanuc	Karel	RoboGuide
Epson	SPEL+	Epson RC+
Nachi Fujikoshi	SLIM	FD On Desk
Adept	V+	Adept ACE

ROS, an open-source project, provides a common framework for robotics applications. ROS is heavily utilized by the research community for service robotics applications, but its technology can be applied to other application areas, including industrial robotics. ROS capabilities, such as advanced perception and path/grasp planning, can enable robotic applications that were previously technically unfeasible or cost-prohibitive. ROS simplifies complex programming tasks by providing a robust and flexible framework where developers and researchers can focus on new algorithm development without coping with trivial low-level hardware communication problems. In terms of flexibility, ROS offers standardized process communication and a variety of drivers for interfacing with different kinds of hardware.

ROS-Industrial basically extends the advanced capabilities of ROS software to industrial robots working in the production process. ROS-Industrial consists of many software packages which can be used for interfacing industrial robots. These packages are BSD (legacy) / Apache 2.0 (preferred) licensed program, which contains libraries, drivers, and tools for industrial hardware. ROS-I architecture can be seen in Fig. 3.2, where in terms of a flexible system, the ROS-I controller layer is one of the most important parts. The ROS-I controller layer contains vendor-specific packages (drivers) for interfacing with different manufacturer-supplied industrial robots. Besides the vendor-specific drivers, another important section is Unified Robot Description Formats (URDFs), which is the standard ROS eXtensible Markup Language (XML) representation of the robot model and is unique for every robot model. Furthermore, URDF is needed by MoveIt! software for motion planning, described in Subsection 3.2.2.

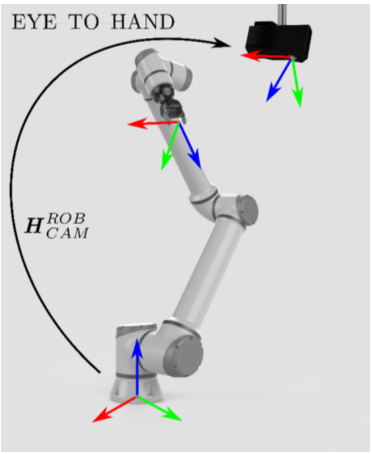


3.2. Fig. ROS-Industrial High level Architecture [144].

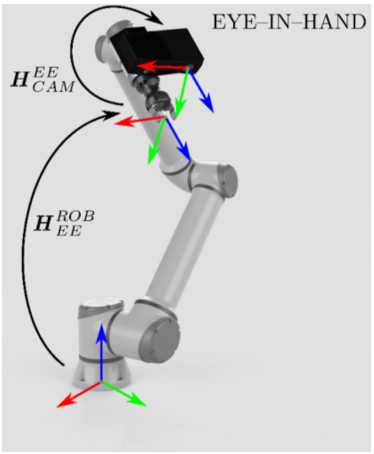
3.2.1. Hand-eye calibration

The calibration process depends on the method of how the camera is attached to the robotic system and mostly depends on the envisioned use case. Two different types of hand-eye calibration can be distinguished:

- Eye-to-hand which is used when the camera is stationary and mounted somewhere next to the robot or above the robot as illustrated in Fig. 3.3(**a**);
- Eye-in-hand which is used when the camera is mounted on the robot, typically as a part of the end-effector as illustrated in Fig. 3.3(**b**).



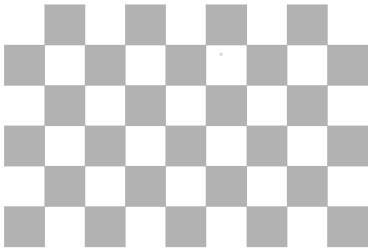
(**a**) Eye-to-hand calibration



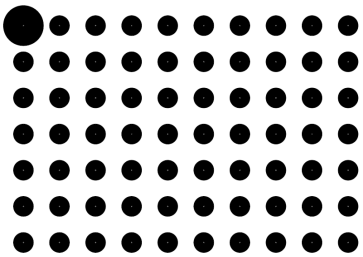
(**b**) Eye-in-hand calibration

3.3. Fig. Hand-eye calibration [145].

Typically, for calibration purposes, an object with specific features is utilized, the so-called calibration target, to efficiently and precisely determine the camera position and orientation with respect to this target. Target can be either 2D or 3D, whereas, for experiments performed in the Thesis, a 2D target is completely sufficient, however, some specific applications could require 3D targets. 2D target typically is similar to the checkers' table with $(x \times y)$ squared as illustrated in Fig. 3.4(**a**). However, the target pattern can be circled as well, as illustrated in Fig. 3.4(**b**). The most important parameter is the distance between square conjunctions or the circle centres, as these parameters are used for pattern recognition algorithms to estimate target pose - position and orientation.



(**a**) Checkers calibration target



(**b**) Dotted calibration target

3.4. Fig. 2D calibration targets.

The calibration equation can be divided into rotational part:

$$\mathbf{R_A R_X} = \mathbf{R_Z R_B} \tag{3.1}$$

, where $\mathbf{R_A}$ - 3×3 rotation matrix;

and translational part:

$$\mathbf{R_A t_X} + \mathbf{t_A} = \mathbf{R_Z t_B} + \mathbf{t_Z} \tag{3.2}$$

, where $\mathbf{t_A}$ - 3×1 translation vector;

Depending on the camera mounting option, the meaning of variables in the calibration equation is changed. In case when the camera is mounted on the robot, **A** is a transformation between the camera and the reference point in the calibration target and **B** is a transformation between the last link of the robot on which the camera typically is mounted. These transformations are known, but **X** and **Z** are unknown, where **X** in this case is a transformation between the last link of the robot and camera, but **Z** is a transformation between robot base and reference point of the calibration target. In case when the camera is static, **X** is a transformation between the last link of the robot and the reference point of the calibration target, but **Z** is a transformation between the robot base and the camera.

3.2.2. Motion planning

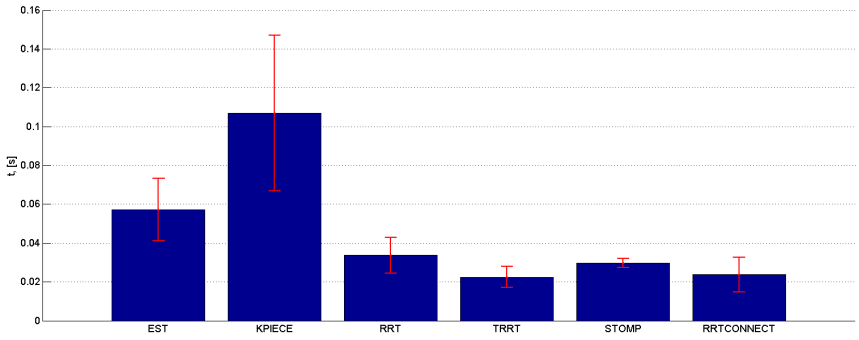
In applications where it is not possible to predefine how the robot must move in order to manipulate objects, for example, in the pick and place tasks where randomly distributed objects must be picked from the container, motion planning more or less is done in real-time or before the corresponding move. Motion planning can be defined as an optimal collision-less trajectory-finding technique how to move the robot from the start state to the goal state while satisfying all the constraints. In order to find a collision-less trajectory, the planning scene must be defined. Stationary objects in the planning scenes can be defined in the URDF robot model, so motion planning algorithms take them into account. Another option is to build *Octomap* from 3D sensors, which can be directly passed into Flexible Collision Library (FCL), the collision checking library that is used by MoveIt! [146].

MoveIt! is a motion planning framework that consists of multiple different packages and tools for motion planning, 3D perception, collision detection, kinematics, robot manipulation and control [146]. Every robot model has a unique URDF and basically, that is the only thing that differs in MoveIt! layer for vendor-specific industrial robot control. The central node in the MoveIt! software is called *move_group* and serves as an integrator: pulling all the individual components together to provide a set of ROS actions and services for users to use [146]. Moreover to use MoveIt! capabilities and communicate with the robot interface nodes MoveIt! configuration package is needed. MoveIt! configuration package is created from URDF robot data by MoveIt! setup assistant GUI and consists of launch and configuration files. Furthermore, in brief, there is no difference between controlled robot specifications and brand. MoveIt! continues ROS concept of flexibility in software architecture.

MoveIt! works with motion planners through a plugin interface. This allows MoveIt! to communicate with and use different motion planners from multiple libraries, making MoveIt! easily extensible. The interface to the motion planners is through a ROS action or service (offered by the move_group node). The default motion planners for move_group are included in the open motion planning library (OMPL) [147] and configured when MoveIt! configuration package is generated. Additionally, stochastic trajectory optimization for motion planning (STOMP) [148] and other motion planners can be easily integrated into a motion planning environment.

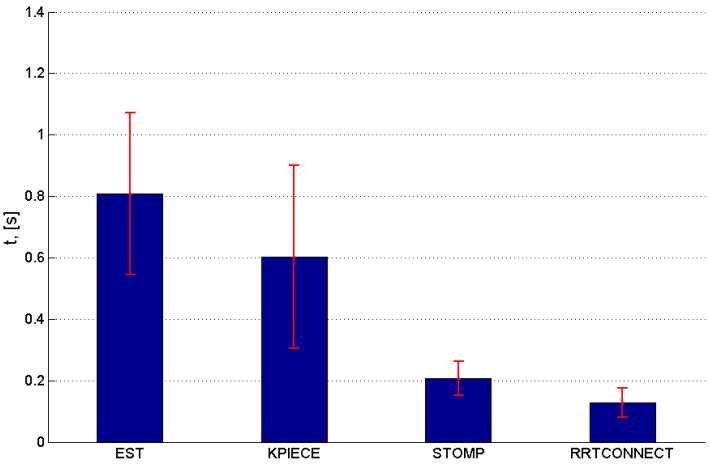
OMPL specializes in sampling-based motion planning, and the probabilistic roadmap (PRM) [149] is among the first sampling-based motion planners. This approach utilizes random sampling of the state space to build a roadmap of the free state space and find the shortest path between the start state and the goal state. Another sampling-based method is creating tree structures of the free state space. This method begins by rooting a tree at the starting configuration of the robot. With the first node of the tree intact, random sampling of the free space then occurs. The planner employs an expansion heuristic, which typically gives the method its name, from which the sample is connected to the tree along a collision-free path. STOMP approach relies on generating noisy trajectories to explore the space around an initial (possibly infeasible) trajectory, which is then combined to produce an updated trajectory with lower cost. A cost function based on a combination of obstacle and smoothness cost is optimized in each iteration [148]

OMPL and STOMP motion planners were experimentally analyzed and compared in order to choose the most appropriate in this experimental system. Motion planners were evaluated using their default configurations in several tests where motions were planned ten times from a fixed start state to a fixed goal state. Motion planners computed trajectories were compared by average planning time, trajectory length and success rate. In total, eight motion planners were experimentally tested, seven of them were part of OMPL and 8th were STOMP. In the first test, motions were planned in a relatively easy scenario without obstacles, with the purpose of eliminating those planners from further tests whose average planning time is significantly higher than other planners' average planning time - Fig. 3.5.



3.5. Fig. Motion planner test in easy scenario [17].

In the second test, obstacles (table and box) were inserted in the planning scene so motion planners had to take that into account and compute trajectories that avoid collisions with these obstacles. Only four motion planners were able to compute valid trajectories in this scenario. The average motion planning time for the second test can be seen in Fig. 3.6. Furthermore, STOMP and RRTCONNECT motion planner trajectories were evaluated by trajectory length and repeatability because their average motion planning time was 3-5 times lower than other motion planners. As a result, the STOMP motion planner was chosen according to computed trajectories and their repeatability.



3.6. Fig. Motion planner test in complicated scenario [17].

3.3. Experimental validation

Experimental validation of the proposed computer vision-based control has been established for simple-shaped-rigid objects in terms of measuring the precision of the estimated grasp pose. The synthetic data generation framework, as described in Section 2.1, can be used in both ways - by generating versatile data for training and also for validation purposes. Such validation is not developed for deformable and transparent as it is unrealistic to define such ground truth positions due to the high variations of object appearances.

A validation pipeline using synthetic data with known ground-truth object position and orientation measurements has been constructed for the measurement and evaluation of systems performance. This computes an optimal reference grasp pose for each predicted grasp and a set of distance or quality metrics comparing the two. The previously described synthetic data generator in Section 2.1 is augmented with object pose (position, orientation) output capability, forming a basis of the validation framework. This can then be used to generate a variety of vali-

dation data sets consisting of scenes with a variety of object types and counts in different lighting conditions and spatial placement. These data sets consist of virtual camera images along with annotation files that contain information such as position, orientation, and type for each object in the scene.

The validation dataset is then used as the input for an inference pipeline employing the data acquisition, object detection, and grasp pose estimation modules to predict a grasp. These grasps are saved to a file along with annotated versions of the input images containing the predicted grasp location and a horizontal projection of its orientation. Inferred grasp data is generated on each pickable object class separately, and the results of each such run are treated independently thereafter. Cases where no pickable objects are found in a scene are also recorded.

To employ a reference grasp comparison method, both ground truth object pose and inferred grasp information must be available. After producing the latter by inference on validation data, the following steps are taken to compute evaluation metrics:

1. For each scene, the inferred grasp is compared with all the ground-truth object positions. Using a heuristic (in this case, minimum Euclidean distance between grasp position and object model coordinate system origin position), the closest object is found. If this is not of the correct type, the output is marked as misclassified and not considered for further quantitative metrics. A data set consisting of each valid grasp pose paired with a target object pose is constructed.

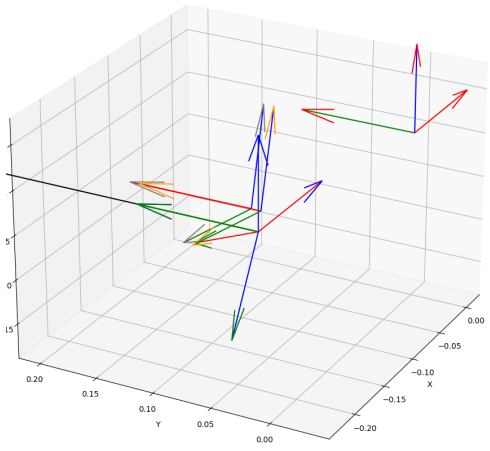
2. The target object and reference grasp are used to compute the nearest optimal grasp according to a heuristic, which varies according to object type and requirements:

    - For cylindrical plastic bottles with a single correct grasp orientation, a reference grasp that lies on its central circumference at the model coordinate system XZ plane is selected. The reference grasp is defined as having the same local Z-axis projection on the target XZ plane as the inferred grasp but corrected for local Y-axis projection deviation from the target Y-axis direction. This heuristic penalizes position errors more than orientation errors resulting therefrom.
    - For metal cans of broadly rectangular cuboid shape, a set of 4 valid grasps is constructed (two for each of the broadest faces, facing in each direction as can long axis direction is not controlled for) by projecting the object model origin onto the corresponding faces and applying the required rotations.

3. A data set containing each grasp paired with the corresponding reference grasp and scene identifier is saved to the file.

The reference grasp computation process is illustrated in Fig. 3.7. Every set of vectors corresponds to a local coordinate system (red, green, and blue being the x-, y- and z-axis unit

vectors after the corresponding rotation and translation transformations are applied). Red arrow tips mark the world coordinate system. Green marks the object (bottle) local coordinate system. Blue marks the object's coordinate system after it has been aligned with the world's horizontal plane by a rotation around the object's long axis (as bottles are cylindrically symmetric). Grey marks the inferred grasp. Orange marks the reference grasp aligned with the object's local zx plane, normal to the object's surface and shifted to be located on its central circumference.



3.7. Fig. A visualization of the reference grasp computation procedure.

In total, 1000 scenes were generated for the evaluation of the estimated grasp poses. The number of objects in each scene varied according to four presets. Each preset contains 250 randomized mixes of bottles and cans. For each preset, the maximum number of objects was defined as follows:

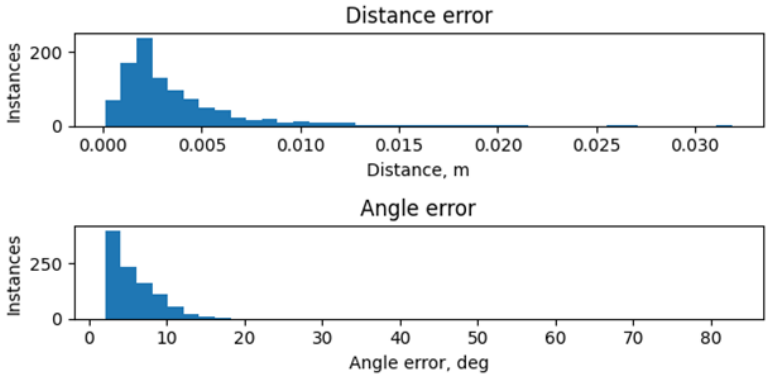
- Preset 1 – 10 objects
- Preset 2 – 20 objects
- Preset 3 – 50 objects
- Preset 4 – 100 objects

The obtained results from the experiments are compiled in Table 3.2. For both object types, the classification results show no errors in the ability to correctly classify the object type. Valid grasps have been acquired in 99.9 % of cases for the object type of bottle and 99.8 % of cases for the object type of can. For this result, some corner cases can be present due to the dynamic features of the environment and especially the randomness of the object positions. In some cases, all the objects can be positioned in a way that doesn't fit the requirements to be recognized as

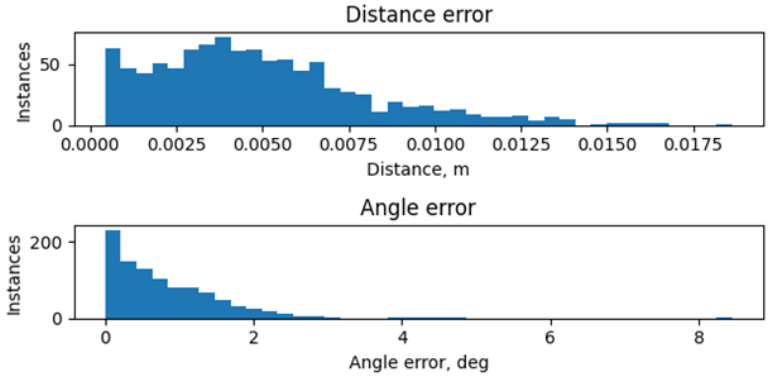
3.2. table

Object detection and grasp pose estimation results

Scene count	Object type	Position error		Rotation error		Valid grasps (%)	Classified correctly (%)
		Mean average (mm)	Stdev (mm)	Mean average (degrees)	Stdev (degrees)		
1000	Bottle	4.926	3.156	0.802	0.736	99.9	100.00
	Can	3.722	3.446	4.637	4.432	99.8	100.00

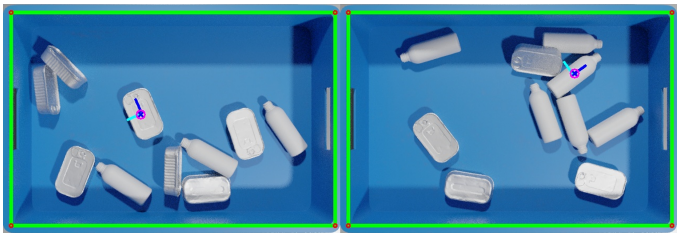


3.8. Fig. Histogram of distance and angle errors for grasp pose estimation of the can.



3.9. Fig. Histogram of distance and angle errors for grasp pose estimation of the bottle.

a valid grasp. In this case, the most accountable limitation is that the grasping approach angle could be in a collision with the container where objects are distributed, and if all the objects are positioned as that, there could be no valid grasps, however, the probability of such circumstance



(**a**) Preset 1 – 10 objects

(**b**) Preset 2 – 20 objects

(**c**) Preset 3 – 50 objects

(**d**) Preset 4 – 100 objects

3.10. Fig. Examples of estimated grasp poses within different presets in the validation process.

is low, which also shows in the results.

The second validation aspect is connected to the precision of the estimated grasp pose. The distance error and orientation error distribution of the achieved results are illustrated in Fig. 3.8 for object-type cans and in Fig. 3.9 for object-type bottles.

On average, for both types of objects, the errors are below the margins of +/- 5 mm for position error and +/- 5 degrees for rotation error. Even if the estimated grasp pose precision was

higher than that, it still was a valid grasp, as the determined point was on the surface of the object where it could be grasped. The only drawbacks of higher error can be connected to the post-grasping steps, e.g. placing of the object and greatly depends on the intended application scenario. For this matter, the validation framework can also determine if an additional repositioning system should be incorporated into the process to ensure the precise placement of the object. Another aspect experimentally evaluated in this section is the ability to find at least one pickable object in the scene, which has shown a success rate of 99.9 % as indicated in Table 3.2. When the system is deployed in a real-life scenario, the differences between synthetic results and results achieved in the real-life setup can be expected. However, due to the nature of randomly dropped objects, it is hardly possible to evaluate the precision equally as with the synthetically generated data. Due to this reason, the evaluation of the real data has been taken by testing the systems' capability to grasp at least one object in every scene. For these purposes, similar tests with different presets were replicated in real life, where the proposed approach succeeded with at least one successful grasp in each scene without errors in the classification task.

3.4. Summary

The chapter on computer-vision-based control takes a significant step forward by leveraging the synthetic data generation approach to enhance robotic capabilities. By integrating computer vision techniques, robots are empowered to "see" and comprehend their surroundings, thereby achieving a higher degree of adaptability in the face of dynamic conditions. A pivotal aspect of this approach lies in the emphasis placed on grasp-pose estimation, where dedicated algorithms are trained exclusively on synthetic data, playing a vital role in detecting and precisely locating objects within the robot's workspace and enabling efficient and accurate object manipulation, particularly in tasks like pick-and-place operations.

The synthetic image generator not only serves as the primary source for training deep-learning models but also takes on a critical role in the validation process. Its versatility allows for the rapid evaluation of various aspects of the proposed algorithms, providing tentative precision results that demonstrate the viability of the specific use case. This capability becomes invaluable in identifying areas that may require further refinement or additional training before transitioning to real-life implementation. Thus, the synthetic data generation approach acts as a powerful tool in the iterative development and optimization of the computer vision-based control system. By combining computer vision-based control with synthetic data generation and efficient motion planning, industrial robots gain higher levels of autonomy and flexibility. This approach expands the usability of robotic systems, making them more accessible to non-experts and capable of handling various tasks in dynamic and uncertain environments.

4. IMITATION-LEARNING-BASED CONTROL

Imitation learning is a broad field of study in its own right that promises to overcome a wide range of challenges, provided that it is possible to obtain a corpus of demonstrations showing how the task is to be accomplished [99]. When dealing with industrial robotics, this is often the case, as the tasks we wish to execute are ones that human operators already routinely perform. Therefore, an imitation learning-based approach is developed for human demonstration incorporation in the knowledge acquisition process for industrial robots. Actions taken by humans in the physical environment are recorded, and a training data set structured in the form of explicit demonstrations is programmatically produced. These are augmented with additional, indirectly observed state variables and control signals. Then, artificial neural network models are trained to reproduce the trajectories therein as motion plans in Cartesian space.

Given the inherent trade-off between task specificity and required model complexity when it comes to observation data modality and model output, motion capture has been selected as a convenient middle ground. It does not require a simulated environment as approaches that involve virtual reality do [112, 116], obviates any issues that may come with having to learn motions in robot configuration space by operating in Cartesian coordinates, and does not necessitate the use of complex image processing layers as found in models with visual input modalities [112, 150]. The main limitations of this approach are that it requires motion capture equipment, which is more expensive than conventional video cameras and Cartesian motion plans need an additional conversion step into joint space trajectories before it is possible to execute them on a real robot. Given the identified challenges during the literature review and the goals of the Thesis - the advantages of compact models, rapid training times, and results that are possible to evaluate ahead of time were deemed to outweigh the drawbacks.

To help guide the development process and serve as a means of evaluation, an example use case was selected – object throwing to extend the robot's reach and improve cycle times. In particular, the task of robotically sorting plastic bottles at a recycling plant was to be augmented with a throwing capability. While on the surface, the task is almost entirely defined by elementary ballistics that can be programmed explicitly, it serves as a convenient benchmark for robot programming by way of demonstrations. The task was deemed sufficiently non-trivial when considered from the perspective of a Markov decision process or time-series model only given limited information about the system state at any given time step, while also being suitable for intuitive evaluation by human observers due to its intuitively straightforward nature. Furthermore, the relationship between release position, velocity, and target coordinates provides an obvious way to quantitatively evaluate model performance against training and validation data – extrapolated throw accuracy. The author's main contribution to this chapter is mainly connected with the conceptualization and methodology, however, the implementation was performed by Peteris Racinskis, thoroughly assisted by the author of the Thesis.

4.1. Preliminaries

In this chapter, an *agent* can be taken to mean the part of the system that acts based on state or observation vectors $\mathbf{s}$ in an *environment*, according to a *policy* π which is specified by a parametric *model* – a function $\pi_{\boldsymbol{\theta}}(\mathbf{s})$ with parameters $\boldsymbol{\theta}$ tuned in optimization and produces an output *action* a. For all practical intents and purposes, this means that references to the agent, model and policy are almost interchangeable in most contexts. A formalism common in both reinforcement and imitation learning contexts is the Markov decision process (MDP), formally given by [99]

$$MDP = (S, A, T, R, I) \tag{4.1}$$

, where S is the set of states s, A is the (formally discrete) set of actions a, $T : S \times A \rightarrow S$ is a state transition function that encapsulates the environment, $R : S \rightarrow \mathbb{R}$ is a reward function associated with each state and $I = p\,(\mathbf{s}_0 \in S)$ represents the initial state distribution. In many imitation learning-related cases, a reward function need not be specified or considered. Moreover, if one is willing to break with the strict formal definition and give up the use of mathematical tools defined only on finite probability distributions, continuous action spaces can also be considered.

One important characteristic of the MDP is that it is history-agnostic – the future state distribution of the system is uniquely defined by its current state. While technically true for physical environments, it is often the case that instead of a complete state representation vector $\mathbf{s}$ this method is instead operating on a more limited *observation* $\mathbf{o}$ (often interchangeably referred to as $\mathbf{s}$ for brevity), where each element o_i is given by some function $f_i(\mathbf{s})$. It is, therefore, possible that historical observations contain information about hidden system state variables, even assuming the MDP formalism holds for the underlying environment. An example of this is the case when individual observations contain only the current position of an object but not its derivatives, as in video data. In such situations, it may prove beneficial to break with the formalism further by redefining the policy to operate on sequences of $k + 1$ previous states/observations.

$$t, k, n \in \mathbb{N}; \pi_{\boldsymbol{\theta}} : \{(\mathbf{s}_n)_{t-k}^{t}\} \rightarrow A \tag{4.2}$$

, which, by allowing for input sequences of variable length, may become a function defined on the entire known state history:

$$\pi_{\boldsymbol{\theta}} : \{(\mathbf{s}_n)_{1}^{t}\} \rightarrow A \tag{4.3}$$

These adjustments allow for the employment of sequence-to-sequence predictor architectures also studied in other areas of machine learning, such as recurrent neural networks (RNNs) [151] and transformers [152]. Finally, if continuous action spaces are permitted, it is no great stretch to also consider formats where the action corresponds to a predicted future state to be used for static motion planning or in a feedback controller.

$$\pi_{\boldsymbol{\theta}} : \{(\mathbf{s}_n)_1^t\} \rightarrow S \tag{4.4}$$

, which, when running the model on its prior output, is equivalent to sequence-building tasks encountered in domains such as text generation. If the sequence of states contains a sequence of joint state vectors (a joint space trajectory) in the form

$$((\mathbf{y}_1, t_1), ..., (\mathbf{y}_n, t_n)) \tag{4.5}$$

, where each *waypoint* $(\mathbf{y}_i, t_i)$ consists of a joint space goal $\mathbf{y}$ and corresponding timestamp t, it could, in principle be used to directly produce control inputs

$$\mathbf{y}(\tau) = \text{interp}(\mathbf{y}_{t_k}, \mathbf{y}_{t_l}, \tau) \tag{4.6}$$

, where $\mathbf{y}(\tau)$ denotes the interpolated, time-varying joint state setpoint to be used in a feedback controller and t_k, t_l are timestamps such that $t_1, t_2, ..., t_k \leq \tau \leq t_l, t_{l+1} ..., t_n$. In the case of models operating in Cartesian space, the joint state vectors need to be found by application of inverse kinematics. However, in practice, just these steps are not sufficient, and the joint space trajectories have to be processed – constraints must be checked for violations, collisions avoided, and optimization of waypoint count, placement and timing may be performed [153]. All of these tasks fall under the umbrella of motion planning.

4.2. Proposed approach

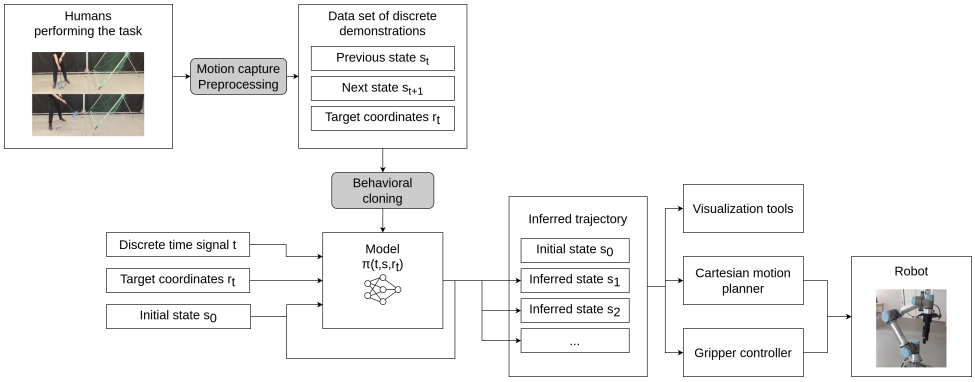
Three crucial decisions need to be made when devising an imitation learning-based approach to robot control:

- data collection – what will serve as the expert policy? How will the data be obtained?
- Model architecture – what type of model template will be used to learn the policy? What are its inputs and outputs, and what pre- and post-processing steps will these formats call for?
- Control method – how will model outputs be used in robot motion planning or feedback control?

Considering that the goal of this chapter is not to examine any of these sub-problems in detail but rather to come up with a holistic approach to combine all of them, trade-offs that affect more than one step at a time need to be considered. The recording of human performances as the expert data set is also a constraint imposed by the objective that was set out to be accomplished. Therefore, when selecting the demonstration data modality, three possibilities were examined:

- raw video – using conventional cameras to record a scene and use these observations as model inputs directly;
- motion capture – obtain effector and scene object pose (position and orientation) data with specialized motion capture equipment that consists of multiple cameras tracking highly reflective markers affixed to bodies of interest;
- simulation – using a simulated environment in conjunction with virtual reality (VR) or another human-machine interface to obtain either pose data as with motion capture – or record the configurations of a robot model directly.

Raw video is the most attractive approach from a material standpoint – it does not involve motion capture or VR equipment, does not require simulated environments, and, in principle, presents the possibility of a sensor-to-actuator learned pipeline if the video data were used directly in the model's input to predict joint velocities. In practice, however, not only does using image data call for the employment of more complex model architectures such as convolutional neural networks, but images are inherently 2-dimensional representations of 3-dimensional space, leading to ambiguities in the data – and one quickly runs into difficult issues such as context translation that confound the issue further [150].



4.1. Fig. A high-level overview of the proposed approach [20].

In contrast, unambiguous pose information is readily obtained with motion capture or simulation. Simulating a robot and collecting data in joint space allows for a simple final controller stage. However, it very tightly couples the trained model to a specific physical implementation, and reasoning about or debugging model outputs obtained this way is not straightforward. Thus, the main comparison is to be drawn between object pose data as obtained in a simulation and with motion capture. Assuming motion capture equipment is available, setting the scene up for collecting demonstrations is a matter of outfitting the objects of interest with markers and defining them as rigid bodies to be tracked. However, there is a degree of imprecision in the

observations, and information about different rigid bodies is available at different instants in time.

Simulation allows recording exact pose information directly at regular time intervals, but setting up the scene necessitates creating a virtual environment where the task can be performed. In addition, for realistic interactions, a physics simulation and immersive interface such as VR are required, but even this does not present the human demonstrator with the haptic feedback of actually performing the task in the real world. Finally, whenever data obtained in a simulated environment is used to solve control challenges, the so-called reality gap or sim2real problem needs to be tackled – the difference between the responses of simulated and real environments to the same stimuli [154]. As such, it was decided that the benefits of a simple setup and inherently natural interaction with the physical scene outweigh the precision and cost advantages of simulated environments, and motion capture was selected as the source of demonstration data. For extracting additional information specific to the throwing task, the object thrown is tracked, and its pose information is used to annotate each observation with target coordinates extrapolated from its flight arc and a release timing signal determined by heuristics described in Subsection 4.3.2 (pre-processing).

The next crucial decision to make is what the input and output spaces of the model will be. Setting aside the additional control parameters (target coordinates and release signal), when provided with a sequence of pose observations, the main options are:

- Pose-Pose – predict the pose at the next discrete time step from the current observation;
- Pose-Derivatives – predict velocities or accelerations as actions;
- Joint state-Joint target – analogous to pose-pose, but in joint space;
- Joint state-Joint velocity – analogous to pose-derivates, but in joint space.

Approaches crossing the Cartesian-joint space domain boundary were not considered, as the model would be required to learn the inverse kinematics of the robot. The advantages of having a model operate entirely in joint space would be seen at the next stage – integration with the robot controller – as motion planning to joint targets is trivial. However, training such a model would require mapping the demonstrations to robot configuration space by solving inverse kinematics on them, much the same as with a model operating in Cartesian space. While computationally less expensive at runtime, this approach is more difficult to reason about or explain, complicating the hyperparameter discovery process. Any model obtained this way would also be tightly coupled to a specific robot. Hence, all models were trained in Cartesian space, with the motion planning step left for last.

Outputting pose derivatives (linear and angular velocities) is advantageous as it enables the use of servo controllers, and input-output timing constraints are less stringent than they would

be in a sample frequency-based model where a constant time step is used to modulate velocities. However, a model-in-the-loop control scheme is required – forward planning is not possible without a means to integrate model outputs with realistic feedback from the environment. Moreover, for training, the model derivatives of the pose need to be estimated from sequential observations, which are highly susceptible to discretization error.

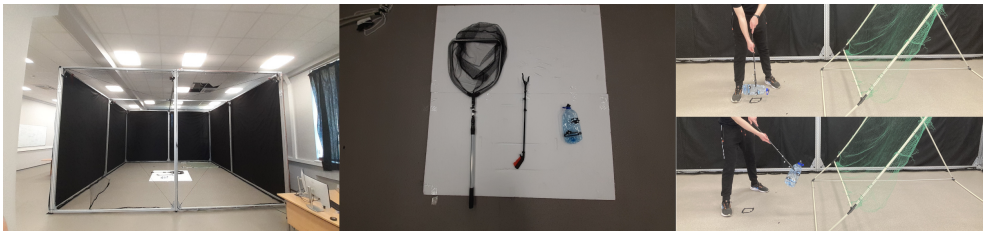
Predicting the future pose allows for using discrete-time pose observations directly, and, assuming a pose following controller can keep within tolerances – which, in this case, were determined by the requirements for triggering gripper release, see Subsection 4.3.4 – forward planning becomes a matter of running the model autoregressively (using its previous outputs for generating a sequence). This approach naturally lends itself to the employment of recurrent model architectures. The main drawback is that pose derivative information is encoded in the time step, which imposes constraints on observation and command timing, complicating the use of such a model in a real-time feedback manner. It also requires that a method for accurate time parametrization of Cartesian motion plans is available.

Ultimately, the approach selected uses Pose-Pose policies with footprints given in Equation 4.16. These predict sequences of states containing position and orientation in Cartesian space. Such sequences can then be used as inputs to Cartesian motion planning algorithms, which produce joint trajectories to control the robot, as described in Subsection 4.1.

A high-level overview of the proposed approach is illustrated in Fig. 4.1. Demonstrations are recorded with motion capture equipment, a split and cropped data set augmented with target coordinates and a gripper actuation signal is created. This is used to train neural network models in behavioural cloning. When operating in the open-loop regime, prior model outputs are used to autoregressively predict subsequent states. The resulting sequence can then be fed into a Cartesian motion planner to control a robot.

4.3. Implementation

The system devised in this section can be broken down into three main sections – the collection of observations in the physical environment (Subsection 4.3.1), a pipeline for turning raw observation data into structured demonstrations with additional control signals (Subsection 4.3.2), and neural network model implementations (Subsection 4.3.3). System performance was evaluated, and design feedback was obtained in two main ways. First, qualitative observations of the generated trajectories were made using spatial visualization in a virtual environment, followed by execution on simulated and real robots (Subsection 4.3.4). When satisfactory performance was attained, a series of quantitative metrics were computed for comparing system outputs with training and validation datasets on different model architectures and hyperparameter sets (Subsection 4.3.5).



4.2. Fig. Motion capture equipment and demonstration acquisition process [20].

4.3.1. Data collection

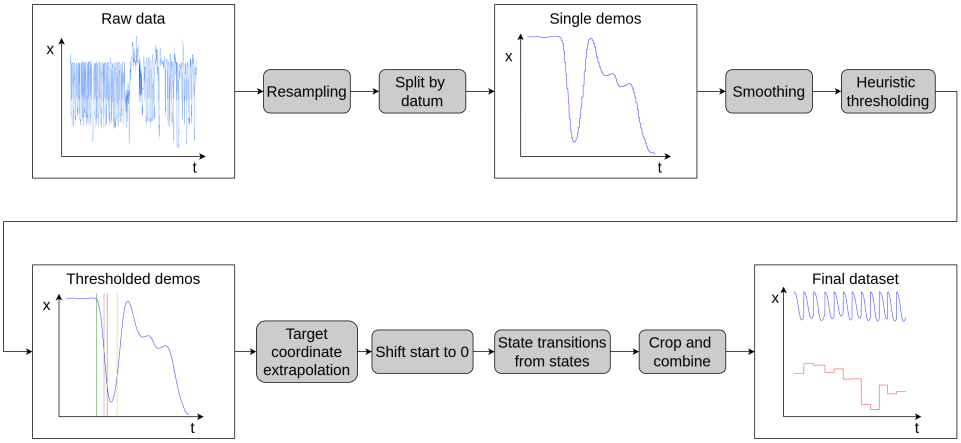
All demonstrations were recorded using *OptiTrack* equipment that consists of a set of cameras, highly reflective markers to be attached to trackable objects, and the *Motive* software package which handles pose estimation and streaming. As shown in Fig. 4.2, a cage with eight cameras installed serves to hide external sources of specular reflections from the cameras and confine moving objects such as drones from leaving the scene. To simulate an industrial robot end effector, a gripping hand tool was equipped with markers. Since the chosen application is for throwing plastic bottles, one such was also marked for extrapolating target coordinates and identifying the release point in a demonstration. A start point is laid down on the floor to serve as a datum for automatically separating the recorded demonstrations during pre-processing. Holding the effector stand-in here for at least a second before each demonstration provides a consistent signal that can be identified programmatically.

Nets were used to catch the thrown bottle and prevent damaging the markers. These were also marked to aid in delimiting the ballistic segment of the thrown object's flight. While in principle, it is possible to detect an object in freefall by observing its acceleration, for trajectory extrapolation, this was deemed too prone to error – some impacts radically change the horizontal component of the object's velocity without breaking its fall, and the acceleration of the empty bottles proved to be affected by air resistance to a significant degree owing to their low terminal velocity.

In software, rigid body definitions were created for all objects to be tracked. These were then streamed on the local network, relayed over ROS topics corresponding to each rigid body using a pre-existing package ROS and recorded into a bag file to be converted into a .csv data set. Over the course of this task, two data sets with roughly 150 and 50 demonstrations each were collected. The first was used in the development process but proved to contain throws beyond the capabilities of the robot hardware used. Therefore, a second data set of throws with less pronounced swings was produced to fit within the working volume of the robot arm. A notable feature of the second data set is that initial orientations were randomized to a much lesser degree ($\pm 30^\circ$) than in the original corpus ($\pm 90^\circ$), and the range of estimated target coordinates for the throws were also more tightly constrained (1.5…1.8m along the x-axis as opposed to 1.6…3.6m),

likely contributing to the better estimated throw accuracy metrics as illustrated in Subsection 4.4.

4.3.2. Pre-processing, extraction of implicit control signals



4.3. Fig. Pre-processing pipeline overview [20].

Fig. 4.3 provides a schematic overview of the main steps involved in the data preparation process. For illustrative purposes, graphs of the effector x-coordinate with respect to time are used, but the final picture also contains the estimated target x-coordinate of each discrete demonstration.

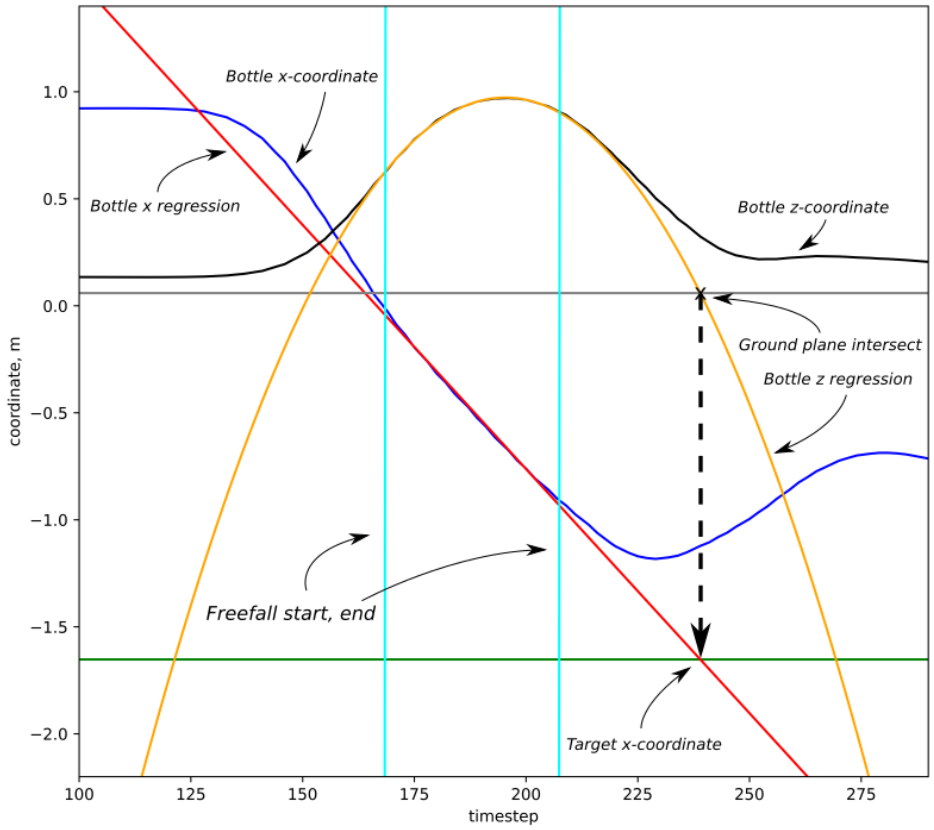
After each recording session, an entirely unstructured corpus is available. It contains timestamped pose observations collected at whatever intervals they happened to have been generated by the tracking software. Each observation only contains information about a single-tracked body. A peculiarity of this particular recording method is that observations about each body tend to arrive in bursts:

$$\mathbf{s}^a_{t1}, \mathbf{s}^a_{t2}, ..., \mathbf{s}^b_{tm-1}, \mathbf{s}^b_{tm} \tag{4.7}$$

where $\mathbf{s}^a_{t\in\mathbb{R}}$ corresponds to the observed state of object a at time t. Thus, to obtain sequential data in discrete time, it is necessary to resample the positions and orientations of all relevant bodies at a constant time step:

$$(\mathbf{s}^a_1, \mathbf{s}^b_1, \mathbf{s}^c_1), ..., (\mathbf{s}^a_k, \mathbf{s}^b_k, \mathbf{s}^c_k) = \mathbf{s}_1, ..., \mathbf{s}_k \tag{4.8}$$

A sampling frequency of $100Hz$ was selected to obtain spatial precision on the order of centimetres given the velocities involved. Fortuitously, the intervals between observed bursts in the actual data are generally in this order as well. Resampling is accomplished using forward fill interpolation, which introduces a slight sawtooth oscillation, but this is removed in the subsequent smoothing step along with other discontinuities.



4.4. Fig. Estimation of the throw target coordinates. Bottle position data is used to annotate each demonstration – consisting of effector pose observations – with throw target coordinates. Regression on the z-axis is used to find the ground plane intersection time, at which extrapolated x- and y-coordinate values are found [20].

After resampling, individual demonstrations are separated using the aforementioned consistent starting position. This may vary from recording session to recording session, so it needs to be identified and configured manually. Specifically, the condition used to identify demonstration start points is as follows:

$$t_{start} \in [t]_1^k : t < t_{start} \wedge (t_{start} - t < \Delta t_{max}) \Rightarrow$$
$$\Rightarrow x_t \in (x_0 - \delta x, x_0 + \delta x), y_t \in (y_0 - \delta y, y_0 + \delta y) \tag{4.9}$$

Which is to say that every observation no more than Δt steps before t_{start} has to lie within $\pm\delta x, \pm\delta y$ of the datum (x_0, y_0). Each demonstration then consists of observations.

$$(\mathbf{s}_{t_{start}}, ..., \mathbf{s}_{t_{start}+steps}), steps \in \mathbb{N} \tag{4.10}$$

The constants $\Delta t, \delta x, \delta y, steps$ are determined experimentally to produce consistent observations for the particular task and may require tuning if the manner in which demonstrations are recorded changes. The newly split demonstrations are saved in separate files, thus allowing demonstrations generated in different sessions to be processed at once.

As further thresholding steps rely on estimates of pose derivatives, a smoothing step is first employed to remove noise and the slight sawtooth oscillation induced by the resampling step. This is accomplished by applying a rolling average kernel to the trajectory. Nevertheless, it was found that any estimate of derivatives would exacerbate what noise there was in the dataset, leading to inconsistent results. So a simplified estimator $\overline{x'_t}$, combining a correlate of the derivative x'_t with a rolling average filter, was used with satisfactory results:

$$x'_t \propto \overline{x'_t} = \sum_{i=t}^{t+m} x_i - \sum_{i=t-m}^{t} x_i \tag{4.11}$$

This was then used to compute threshold functions corresponding to the start of motion, actuator release and the unfettered freefall of the bottle:

$$f_{moving}(t) = \begin{cases} 0 \text{ if } \forall u < t, \| \overline{\left[\mathbf{r}_{Effector}(u)\right]'_u} \| < \overline{v}_{moving} \\ 1 \text{ otherwise} \end{cases} \tag{4.12}$$

$$f_{release}(t) = \begin{cases} 0 \text{ if } \forall u < t, \overline{\left[\|\mathbf{r}_{Bottle}(u) - \mathbf{r}_{Effector}(u)\|\right]''_u} < \overline{a}_{release} \\ 1 \text{ otherwise} \end{cases} \tag{4.13}$$

$$f_{freefall}(t) = \begin{cases} 0 \text{ if } \forall u < t, \overline{\left[\|\mathbf{r}_{Bottle}(u) - \mathbf{r}_{Effector}(u)\|\right]'_u} < \overline{v}_{freefall} \\ 1 \text{ otherwise} \end{cases} \tag{4.14}$$

, where $\mathbf{r}$ corresponds to the position component of the observation vector. Note that in the first case, the norm of a vector derivative is used, whereas in the subsequent two, it is the scalar derivative of a vector norm. As discussed in 4.3.1, the other terminating condition for the freefall segment of the bottle's trajectory is determined using the position of the net:

$$f_{passed}(t) = \begin{cases} 0 \text{ if } \forall u < t, x_{Bottle}(u) \geq x_{Net}(u) \\ 1 \text{ otherwise} \end{cases} \tag{4.15}$$

The particular contents of the thresholding functions used are application-specific, but the method of detecting discrete events based on relative and absolute derivatives should prove broadly applicable. As with the split step, constants for thresholding were determined by way of

inspection and would certainly be unique to each type of task. The output of $f_{release}$ serves as the gripper actuator signal estimate. f_{moving} is used in the final align, crop, and combine steps to determine the first observation to include in the dataset. $f_{freefall}$, f_{passed} are used in estimating the desired target coordinates of each throw to give models the capability to be aimed. This is done by applying quadratic regression to the z-coordinate of the object thrown, finding its intersection with the ground plane in corresponding x and y-coordinates – a geometric illustration of this process can be found in Fig. 4.4. The actuator signal and target coordinates are concatenated to each observation vector, with target coordinates being constant within each demonstration. A time signal is also added to each observation, as this was found to improve the performance of feedforward models. Finally, when combining the demonstrations into a training data set, all position vectors are shifted so that the start of motion corresponds to the origin of the coordinate system. This way, the models are trained to operate relative to the effector starting position.

4.3.3. Models

Two classes of parametric models were studied as part of this task – simple feedforward neural networks and RNNs, operating autoregressively. The choice was motivated by two factors:

- the dynamics of the problem were deemed to be simple enough that even small models would be able to model them adequately – making it possible to quickly train on development machines with low parameter counts, using pre-existing code libraries;
- given the recent advances in employing sequence-to-sequence models for imitation learning tasks, it was decided that models with broadly similar footprints and characteristics should be used to enable further research in this direction.

After an initial hyperparameter discovery process, the values in table 4.1 were arrived at for both model types respectively. A range of model sizes and training epoch counts was compared for both models. In the case of the recurrent neural network, performance with different learning rates was also evaluated. As the feedforward models were developed first, performance with and without a time signal in the input data was also compared. For the RNN architecture, a gated recurrent unit (GRU) was selected as prior research suggests that it outperforms long short-term memory (LSTM) when dealing with small data sets of long sequences [155].

The basic model footprint is given as follows:

$$(\mathbf{r}_{t+1}, \mathbf{q}_{t+1}, g_{t+1}) = \pi_{\boldsymbol{\theta}} \left(\frac{t}{f_{sample}}, \mathbf{r}_t, \mathbf{q}_t, g_t, \mathbf{r}_t^{target} \right) \tag{4.16}$$

, where $\mathbf{r}_t, \mathbf{q}_t, g_t$ represent the end effector translation vector, orientation quaternion and gripper actuator signal, respectively at time step t in both the input and output. The input is augmented with a time/phase signal (discrete time step divided by the sampling frequency, in this case,

4.1. table

Model hyperparameters [20]

Parameter	Feedforward	RNN
Architecture	2 dense hidden layers, ReLU	GRU, dense linear output
Parameter counts	128-1024 perceptrons per layer	128-512 perceptrons in the unit
Training epochs	20-100	300-1200
Batch size	64	32
Optimizer	Adam	Adam
Learning rate	10^{-4}	$10^{-3}, 10^{-4}$

$100Hz$) relative to the start of the demonstration or generated trajectory, as well as the target coordinate vector $\mathbf{r}_t^{target}$ which corresponds to the extrapolated target coordinates in the demonstration data set and commanded throw coordinates at inference. The time signal was added to the input as it was found that trajectories generated by feedforward networks were liable to diverge without it, and the data sets thus modified were used for all training thereafter. The gripper control signal has values $\{0, 1\}$ in the training data set.

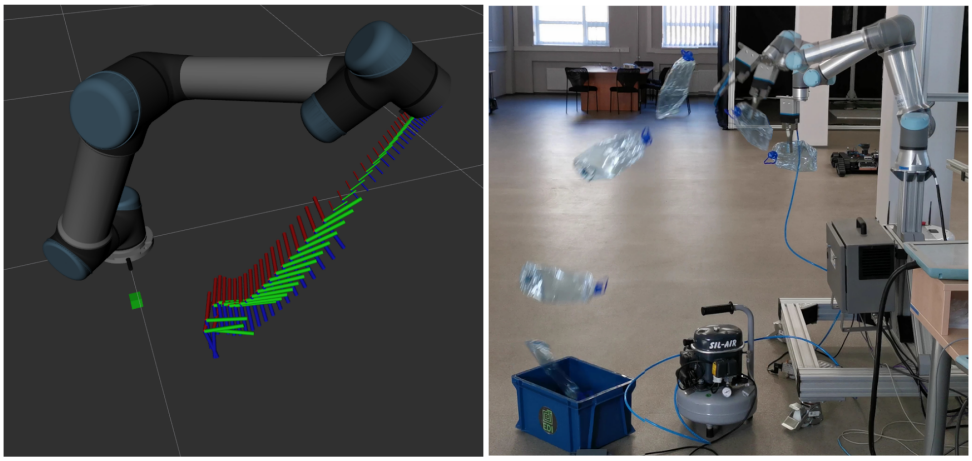
For training both types of networks, state transitions $\left(\left(t, \mathbf{s}_t, \mathbf{r}_t^{target}\right), \mathbf{s}_{t+1}\right)$ are constructed. In the case of the feedforward network, these state transitions are then shuffled and batched independently. For the RNN, pairs of feature-label sequences are formed corresponding to complete demonstrations, and the loss function is computed on the entire predicted output sequence. To hold these variable-length sequences, a ragged tensor is used, which is batched along its first axis and ragged along the second. With both types of networks, the Huber loss function is utilized.

4.3.4. Visualization and execution

As discussed in the introduction, an important part of the reason for selecting throwing as the application was the fact that it is easy for humans to judge the qualitative performance aspects of this task. To accomplish this, a means of visualizing the model outputs was required. When operating in open-loop mode (without interfacing with the physical or simulated environment – running the model on its previous outputs), it is possible to precompute trajectories and simply save them as sequences of robot states. To aid in estimating whether these trajectories were feasible, a tool for visualizing these sequences in the robot coordinate system was developed (Fig. 4.5a).

Given a policy that synchronously predicts the state of the system at the next time step, there are multiple possible ways to use it in robot control. The simplest approach is static trajectory planning – precompute a sequence of states and plan the motion between them. This is somewhat hindered by the lack of existing tools for precise time-parametrized Cartesian path planning in the

ROS ecosystem. An alternative is a real-time pose following a servo controller. The advantage of the latter approach is that closed-loop control is possible – with state observations taken from the environment, potentially allowing the model to compensate for offsets in real-time. However, this presents the challenge of tuning controller gains. The method that was ultimately employed was open-loop planning of Cartesian paths – with timing approximated through a combination of time-optimal trajectory generation [153], followed by scaling the trajectory to the correct total duration. To control the gripper, a threshold value was set, the point in the trajectory at which this value was crossed was found, the corresponding joint state was computed, and a callback function was set up to trigger when this joint state was reached within an angular tolerance of 0.05 radians for each joint independently.



(a) Visualization of the inferred trajectory in *rviz*.

(b) Throw execution on a *Universal Robots UR5e*.

4.5. Fig. Qualitative performance evaluation – visualization and execution. In both cases, the trajectories were sequences of pose (position, orientation) goals obtained by running the models autoregressively on their outputs at prior time steps. Release timing is represented in (a) by the change in marker size [20].

While this is adequate for evaluating the positioning of the generated trajectories and approximates the velocity profile closely enough for successful throws to be executed (Fig. 4.5b), a fully-featured time-parametrized Cartesian path planner or correctly tuned pose-following controller would be required to judge the throwing accuracy on real hardware and generalize the proposed approach to other tasks. However, the development of such general-purpose tools was deemed to be outside the scope of this task, the focus of which is on the collection of demonstration data and imitation learning methods.

4.3.5. Evaluation metrics

To draw comparisons between models of different architectures developed as part of this task, trained with differing hyperparameter sets, quantitative evaluations needed to be computed. As this is not a standard task with agreed-upon metrics and benchmarks, some exploratory work was required to arrive at quantifiers that agree with intuitively self-evident characteristics. While, in principle, it should be possible to judge models based on throw accuracy, this is not very helpful in the early stages of research when most models fall short of attaining the desired objective. Furthermore, limitations imposed by the aforementioned time parametrization issues with Cartesian motion planning in ROS make such a comparison as yet infeasible.

Hence, it was decided to compare model outputs with the demonstration data set. As judging against the training data set is only informing to the extent to which the model has been able to overfit, a smaller validation data set was set aside for out-of-distribution comparisons. The outputs to be compared were obtained by executing the models autoregressively on each demonstration's initial state for as many steps as were present in the corresponding recorded demonstration. This can be formally stated as

$$\mathcal{D}_{eval} = \left((\tau_1^d, \tau_1^g), ..., (\tau_k^d, \tau_k^g) \right) \tag{4.17}$$

$$\tau^d, \tau^g = (\mathbf{s}_1^d, ..., \mathbf{s}_m^d), (\mathbf{s}_1^g, ..., \mathbf{s}_m^g); \mathbf{s}_1^d = \mathbf{s}_1^g \tag{4.18}$$

, where $\mathcal{D}_{eval}$ refers to the evaluation data set of a single model with respect to either the validation or test demonstration set, but τ_i^d, τ_i^g are the demonstration and generated trajectories (sequences of state observations s_j) sharing the same initial state respectively.

Three broad classes of evaluation metrics were computed: data set-wise (global), step-wise and throw parameters. The first consists of vector similarity measures such as Pearson's correlation coefficient, cosine similarity and distance metrics applied to the entire data set and the trajectories generated against it as concatenated vectors:

$$f_{global}(\mathcal{D}_{eval}) = f : (\tau_1^d, ..., \tau_k^d) \times (\tau_1^g, ..., \tau_k^g) \rightarrow \mathbb{R} \tag{4.19}$$

The second class involves applying various measures to the observation/state variables at each time step:

$$f_{stepwise}(\mathcal{D}_{eval}) = f : \left[\sum_{\tau^d, \tau^g \in \mathcal{D}_{eval}} \sum_{i=1}^{m} \left(g : \mathbf{s}_i^d \times \mathbf{s}_i^g \rightarrow \mathbb{R} \right) \right] \rightarrow \mathbb{R} \tag{4.20}$$

, where the inner function g corresponds to metrics such as position error, rotation error (quaternion angular distance), or categorical cross-entropy in the release signal. Finally, a release error metric was computed trajectory-wise. To do this, the release point of each trajectory was found, the corresponding position found and the velocity vector estimated. When considering ways to

combine these two terms into a single quantitative error estimator, it was decided that a simple ballistic extrapolation and resulting miss distance along the ground plane would serve as a decent first-order approximation of throw accuracy *vis-à-vis* the training or validation data set.

Specifically, the release position $\mathbf{r}_0 = (x_0, y_0, z_0)$ and estimated release velocity $\mathbf{v}_0 = (v_{0x}, v_{0y}, v_{0z})$ were used as parameters in the equations:

$$x(t) = x_0 + v_{0x}t \tag{4.21}$$

$$y(t) = y_0 + v_{0y}t \tag{4.22}$$

$$z(t) = z_0 + v_{0z}t - \frac{g}{2}t^2 \tag{4.23}$$

, where $g \approx 9.81 \frac{m}{s^2}$ - the value of the acceleration of gravity at the Earth's surface.

The ground plane intersect time $t_{intersect}$ was found by setting Eq. 4.23 equal to the ground plane coordinate z_{target} and finding the positive root. Then, ground plane intersect points were found as

$$\mathbf{r}_{intersect} = (x(t_{intersect}), y(t_{intersect}), z(t_{intersect})) \tag{4.24}$$

, and the trajectory-wise throw error metric computed by:

$$f_{throw}(\mathcal{D}_{eval}) = \frac{1}{k} \sum_{\mathcal{D}_{eval}} \|\mathbf{r}^g_{intersect} - \mathbf{r}^g_{intersect}\| \tag{4.25}$$

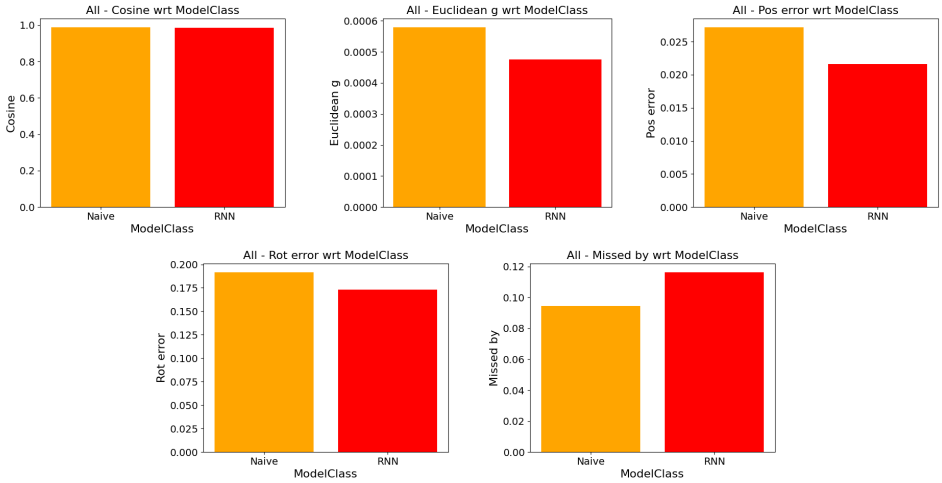
, with $\mathbf{r}^d_{intersect}, \mathbf{r}^g_{intersect}$ being the demonstration and generated throw intersect positions respectively. Seeing as not every model would output a release signal that crossed the threshold (0.5) for every trajectory, this error term was set to be infinite in cases when no throw was defined.

4.4. Results

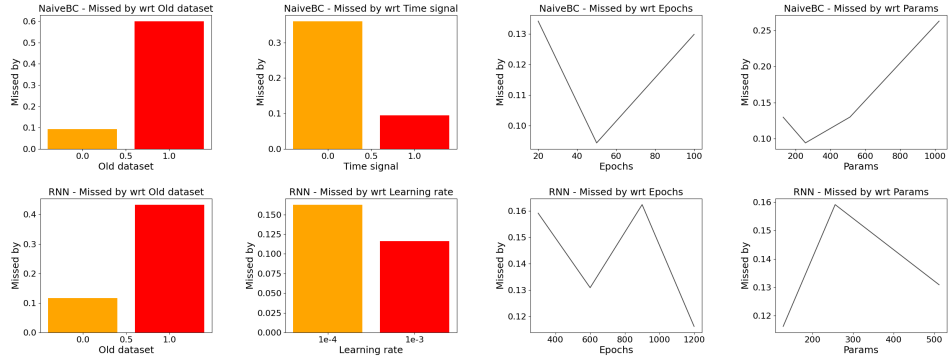
Numerous models were trained on both of the data sets described in Subsection 4.3.1, in both of the architectures discussed in Subsection 4.3.3. After the initial trial and error process, a systematic training regimen produced the following:

- feedforward networks – a total of 48 models, varying the training data set, presence of a time signal in the features, perceptron counts per hidden layer, and training duration;
- RNNs – 36 models with data set, learning rate, model size, and training length being the variable parameters.

Fig. 4.5 illustrates how model outputs were visualized to use their qualitative aspects in guiding hyperparameter choice, and a preliminary implementation on a real robot was achieved – executing trajectories generated by the simpler feedforward network, with target coordinates

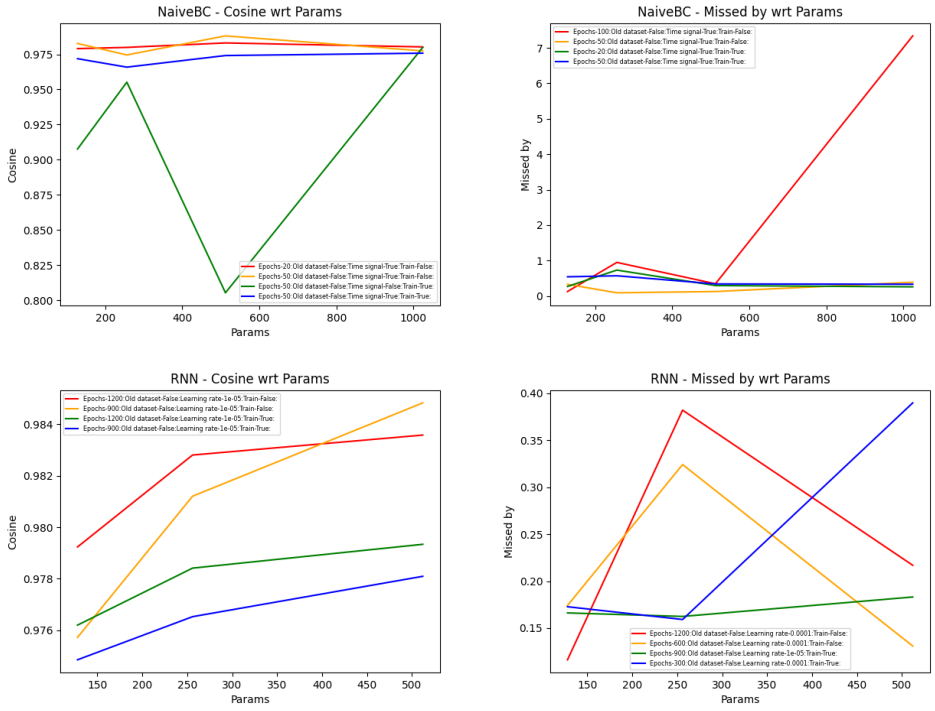


4.6. Fig. Results – comparison between model architectures (*Naive* – refers to the feedforward architecture, also "naive behavioral cloning"), the best performance attained in each class. Top row metrics – cosine similarity (dimensionless), Euclidean distance (all features, no specific unit); mean position error (stepwise, meters). Bottom row – mean rotation error (stepwise, radians), mean throw error (meters) [20].



4.7. Fig. Results – best mean throw error performance for each hyperparameter value, in units of meters. The top row shows feedforward model results, the bottom – RNN [20].

sampled from normal distributions corresponding to their values in the training data set. As can be seen, when these outputs were used to generate motion plans for a Universal Robots UR5e robot equipped with a pneumatic gripper, a plastic bottle was successfully thrown into a target container located outside the robot's reachable volume. Adequate output sequences were attained with both model architectures at multiple hyperparameter combinations, so to draw specific conclusions the quantitative evaluation metrics discussed in Subsection 4.3.5 were



4.8. Fig. Results – cosine similarity (dimensionless) and mean throw error (meters) metrics for feedforward and recurrent models, only varying in the perceptron count parameter (others held constant) – best 2 on train and validation sets each [20].

produced for the 84 systematically trained models described above. Of the feedforward models, 18 out of 48 (37.5 %) successfully triggered a gripper release on every evaluation trajectory, enabling mean-throw error estimates to be computed. Among the RNNs, this was true for 24 out of 36 (66.7 %).

Fig. 4.6 shows a broad comparison between the RNN and feedforward model architectures, with each of the evaluation metric classes being present, though not all the specific metrics discussed – as there was found to be a considerable amount of duplication in their findings. All figures shown correspond to the best result attained by either type in each of the performance indicators, irrespective of other variables.

Fig. 4.7 elaborates upon each model class, showing the best-attained results at various hyper-parameter values. In the case of the feedforward model, the input observations did not initially contain a time signal, which was rectified early on. In the case of the RNN, deviating from the default learning rate was found to be beneficial. For both architectures, the best performance at each epoch count and parameter count is also plotted. Fig. 4.8 shows the impact of only varying

the crucial size (perceptron count) – in each case, the two best-performing models at each fixed parameter combination are selected, and their performance at different values of the independent variable is graphed. In both cases, the smaller, newer dataset with less variance in orientation and target coordinates results in better throw evaluations. Introducing the time signal significantly improves feedforward model performance, whereas, for the RNN, improved results could be attained at higher learning rates. In the case of the feedforward model, training for too long and making the model too large initially appears to be detrimental, while for the RNN, no clear trend was observed in this respect.

The qualitative aspects of the achieved results – the visual representations of trajectories and their characteristics when executed on a real robot – closely resemble throwing motions as executed by the human experts, suggesting that the selected approach to data collection is adequate for encoding the key features of this task. The same can be said for the fact that a large minority of the feedforward models and a majority of the recurrent ones were able to reproduce a complete throw trajectory for every set of initial conditions in the training and validation data sets. The most important takeaway regarding this aspect of the proposed approach is that augmenting the data with time-step information is important to achieve good performance.

Comparing the two proposed model architectures by their best-attained values in terms of distribution similarity measures such as cosine similarity and Euclidean distance, the results are generally quite good for both, with the RNN having the edge in most distance measures but notably one of the feedforward models demonstrating the highest cosine similarity. In terms of step-wise metrics, RNN models typically show better results – which is perhaps to be expected, given that their inputs contain all previous states in the trajectory rather than a single observation.

The same is true for the throw error metric in general, and it is quite apparent when comparing the percentage of valid throw trajectories (ones where actuator release has been commanded) that RNNs have an overall easier time learning the release timing aspect of the task. However, the best single result was attained by an outlier feedforward model – an average error of around 0.09 m, as opposed to around 0.11 m for the best recurrent model – both on the validation data taken from the corrected, smaller demonstration collection (see 4.3.1). This result should be taken with a grain of salt, though, considering the small size of the validation data set (5 demonstrations, set aside from a collection of 45). The performance of the same feedforward model compared against the remaining training data set is much worse – an error of 0.57 m – compared to the best RNN model, which retains an error of 0.26 m. Moreover, at other hyperparameter combinations, RNN models can be seen to retain a sub-0.2 m error on both data sets (such as a 256-perceptron model trained for 300 epochs at a learning rate of 10^{-3}, which attains 0.16 m and 0.18 m error on the training and validation data sets, respectively).

An important thing to note is that this throw error estimate should not be taken as equivalent to a simulation or an actual throw – as has already been discussed in 4.3.1, the terminal velocities of the bottles thrown are low enough to affect the observed trajectories, and in any case, instan-

taneous velocity vector estimates derived from sequences of discrete position measurements are bound to have a degree of error. This is further exacerbated by the fact that the interactions between the gripper and the work object would impart some delay between the release command and full separation. Nevertheless, this estimator does model a large part of the non-linear relationship between the parameters that define a throw – release position and velocity vectors – in a way that is likely to explain a large degree of destination variance among throws performed in the physical world. It is reasonable to assume that lower values of this error term would be strongly correlated with higher accuracy when deployed and tested on physical hardware.

Regardless of model type and parameters, the observed performance on the older data set, uncorrected for robot working volume – with a much greater variance in throw shape, timing, starting orientation, and target coordinates, but only marginally greater size – is significantly worse. None of the feedforward networks attain a throw error estimate under 0.6 m, while RNNs bottom out at around 0.4 m. In the case of feedforward networks, augmenting the input vector with a time signal showed qualitatively observable advantages, and these are also present in the numeric evaluations. An apparent trend of better results being obtainable with smaller models that aren't trained for too long also exists, however, for the reasons discussed above, this may well be spurious. Certainly, no such conclusion can be confidently drawn with respect to the RNNs' performance, as comparatively high performance is achieved by some hyperparameter combinations at every scale explored. A slight but consistent edge in performance was attained by increasing the learning rate of the Adam optimizer from the default value of 10^{-4} to 10^{-3}.

Observing the impact of hyperparameters in isolation yields the clearest insights in the cases of global distribution similarity measures (in the graphs shown, cosine similarity). In feedforward models, a high similarity is attained even at the lowest parameter counts, and making the models larger does not yield unambiguous improvements. Notably, in some settings, there is a considerable degree of variance in this metric. Among RNNs, the perceptron counts examined show a slightly yet still monotonically increasing trend, suggesting that performance gains at larger model sizes may still be made. When it comes to throwing accuracy estimates, neither model architecture shows a clear response to perceptron count, and the feedforward architecture again exhibits outlier results. Interestingly, multiple RNN configurations show initially declining performance on validation data with increased model size, followed by an improvement – potentially evoking the double descent phenomenon [156]. However, there is still a comparable degree of unexplained variance in the results, which forces to be cautious in proposing any concrete explanations.

4.5. Summary

The original goal set out to accomplish was to devise a framework for recording demonstrations by human actors and using these to execute an object-throwing task as a stand-in for

a variety of similar future applications. By using motion capture as the data collection mechanism, it proved possible to employ a series of analytic pre-processing steps to turn raw recording data into demonstration data sets suitable for training two varieties of artificial neural networks – feedforward and recurrent. The models were structured to be able to operate either in-the-loop or autoregressively as forward planners, and it was this latter approach that was further explored – tools were developed for their visualization (useful in hyperparameter discovery) and prototype motion planning software that enable execution on real robots. Model outputs were used to successfully perform throws in the physical world.

The data collection step with motion capture equipment proved to be straightforward to customize for the throwing task. However, it did require some attention to how the physical demonstrations were performed – with allowances for automatic demarcation of distinct demonstrations. A pre-processing pipeline had to be developed to extract these separate trajectories and only their relevant segments, as well as infer hidden state variables through indirect observation. While the latter is likely going to be specific to each application, the methods herein can be trivially adapted so long as these can be expressed as n-th order derivatives of absolute or relative pose variables of tracked objects in the scene. Admittedly, some work, such as target coordinate extrapolation, was entirely specific to this task.

In contrast to some previous work that has explored motion capture in imitation learning, a greater emphasis was placed on using general-purpose machine learning methods – artificial neural networks – to generate the trajectories at inference, as opposed to simple neighbour methods [117] or highly domain-specific approaches grounded in the use of dynamic motion primitives [118]. To more objectively evaluate the performance of the policies obtained with various hyperparameters and on different data sets, a series of quantitative metrics are proposed in this chapter. In general, it can be said that according to these, recurrent networks operating on complete state histories outperform simple deep neural networks operating in a Markovian regime – which is perhaps unsurprising, given the recent successes achieved in applying general-purpose sequence-to-sequence learning methods to the imitation learning domain [157]. As the model precision comparisons rely on indirect estimates rather than empirical data, it is hard to make a direct comparison with state-of-the-art performers in throwing tasks such as [158] – but it should be noted that they use learning at a higher level – outputting parameters that describe each throw – and do not use model outputs to generate Cartesian motion plans directly as proposed here. This makes their entire approach inherently more coupled to a single type of task than this method.

5. EXPERIMENTAL SETUP AND DEMONSTRATIONS

The developed methods within the Thesis are integrated into several demonstrations serving both as showing the practical use of them and as the final demonstrations in three Horizon 2020 projects, namely

- "Artificial Intelligence for Digitizing Industry" (**AI4DI**) described in Section 5.2,
- "Vision, Identification, with Z-sensing Technology and key Applications" (**VIZTA**) described in Section 5.3,
- "Digital Technologies, Advanced Robotics and increased Cyber-security for Agile Production in Future European Manufacturing Ecosystems" (**TRINITY**) described in Section 5.4.

The experimental setup described in the upcoming section, however, serves as the basis for all the listed demonstrations.

The developed methods are integrated into practical solutions, addressing manufacturing challenges and several other important aspects. However, it is also important to mention more of a traditional way of accomplishing tasks that the Thesis tends to improve. Essentially in the majority of cases, a human worker is being substituted, whether in data acquisition as with the synthetic data generation or handling process of arbitrarily placed objects as in all of the demonstrators. In some scenarios within a computer-vison-based control for addressing the sorting of randomly distributed objects (bin-picking), alternatives exist and have been used in the manufacturing sector for several decades that have already substituted human workers. Arguably, the most popular approach is sorting with vibration. Even though such vibro-sorting machines have proved their viability they come with a limited range of flexibility which is a critical factor in a agile production environment when the product variety is keen to change rapidly.

Vibro-sorting machines are best suited for sorting and orienting a specific range of objects with similar properties, such as consistent size and shape. They may not be as flexible when dealing with a wide variety of object types and sizes. Bin-picking is more flexible and adaptable to handle a broader range of objects, even if they vary in size, shape, or orientation. The computer vision system allows the robot to recognize and adapt to different objects within the same workspace. Another important aspect is delicacy and fragility as vibro-sorting machines may not be suitable for delicate or fragile objects as the vibrations and mechanical forces can cause damage or deformation. Bin-picking can be more gentle and precise, making it better suited for handling delicate objects. The vision system can identify fragile items and adjust the gripper's force accordingly to avoid damage.

Another alternative approach in general is traditional industrial robot control, however in this case exact and precise knowledge about the environment is required for the motions even in a simple pick-and-place process. The motions are typically programmed in a non-flexible manner

by human experts and are thus unable to deal with even small environmental variations. The proposed approaches are improving these aspects by being able to perceive a dynamic environment and act in it, generate feasible collision-less trajectories and utilize human demonstrations in learning new tasks, therefore addressing both the maintainability and usability aspects.

5.1. Hardware and software building blocks

The experimental setup is built in a laboratory environment in EDI and within the respective hardware and software infrastructure. The hardware and software infrastructure except for the high-performance computer (HPC) has been carefully researched and mostly built during the development of the Thesis, including the digital versions of it. The main hardware and common software components used in the experimental setup and demonstrations are described in the following subsections.

5.1.1. Hardware components

Grippers

Choosing the right gripper for object manipulation is one of the main prerequisites for the successful automation of pick-and-place tasks. There are many parameters that should be considered, but usually, the task itself mainly determines the gripper type. The two main gripping strategies are vacuum gripping and parallel gripping. Both gripping strategies are included in the hardware infrastructure of demonstrators, specifically for the vacuum gripping Schmalz Rob-Set UR [159] is used, which can be seen in Fig. 5.1(**b**). Additionally, a high variety of different vacuum caps are also included in the hardware infrastructure, which allows rapid testing when different kinds of object material are introduced in the pick and place task. For parallel gripping Robotiq two-finger gripper 2f-140, with a stroke of 140mm, grip force up to 125N and payload of 2.5kg as illustrated in Fig. 5.1(**a**) is being utilized.

(**a**) Robotiq two-finger gripper 2f-140 (**b**) Schmalz Rob-Set UR

5.1. Fig. Robotic grippers.

Depth sensors

The information about the dynamically changing environments is acquired using depth sensors, specifically, the grasp pose is estimated of randomly dropped objects where both pose and orientation can vary in three dimensions. The hardware infrastructure for demonstrators contains two depth sensors that vary in parameters like precision, field of view, working range, weight, dimensions etc. The wide variety allows to adapt and test the systems in different situations, whether the camera needs to be attached to the robot or mounted statically. For example, the common point precision of a Zivid One M depth camera in perfect conditions is around 60 micrometres. The Zivid One M stereo camera that can be seen in Fig. 5.2(**a**) uses structured light as 3D technology, resolution of 1920x1200 operating up to 12 frames per second. The interface of this stereo camera is USB 3.0 SuperSpeed. The Zivid camera is ideal for high precision tasks when the camera can be mounted statically as the high dimensions (226mm x 86mm x 165mm) and weight (2kg) of this camera would restrict some movements for the industrial robots if mounted on the robot.

(**a**) Zivid one M stereo camera

(**b**) Intel RealSense Depth Camera D415

5.2. Fig. Depth cameras.

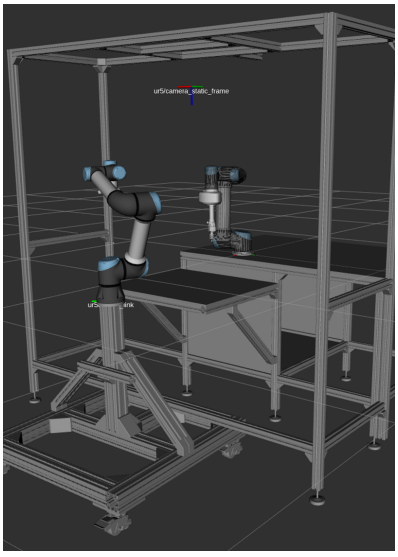
If the task requires mounting the camera on the robot then Intel RealSense depth cameras are a perfect fit as the dimensions (99mm x 20mm x 23mm) and weight (0.073kg) of the camera are considerably lower than the Zivid therefore the camera can be easily mounted or even integrated in robot EoT without blocking the robot movements. The main parameters of the D415 stereo camera illustrated in Fig. 5.2(**b**) are the following: Resolution 1280x720 for depth image and frame rate up to 90fps, point precision from one to three millimetres depending on how far the object from the camera is. Even though the precision is lower than the Zivid camera, the dimensions and weight of the RealSense cameras play an important role in cases when the camera needs to be mounted on the robot.

Industrial robots

One of the main hardware components of the experimental setup and demonstrations is the industrial robots. Industrial collaborative robots (Universal Robots UR5 [160] and UR5e) are used in demonstrators. These robots are ideal for automating low/medium-weight processing tasks. It has a maximum payload of 5kg, an 850mm reach radius, which can be improved with

gripper modifications, and a pose repeatability of +/- 0.1mm for UR5 and +/- 0.03mm for UR5e. Nevertheless, as the demonstrators are built upon ROS, the robot control modules can be easily adjusted for usage with different kinds of robot manufacturers and respective industrial robots. Both the robots and respective environment elements are illustrated in Fig. 5.3(**a**), the 3D models, of robots and environmental elements, crucial for experiments in simulations and also for motion planning are depicted in 5.3(**b**).

(**a**) UR5 and UR5e and the workspace



(**b**) 3D models of UR5 and UR5e robots

5.3. Fig. Industrial robots.

Processing units

For inference purposes, a total of three processing units have been set up and configured for running the demonstrations. Firstly each of the industrial robots has its own dedicated control PC with Ubuntu 18.04 system and running ROS melodic. The Robot Control PCs are set up with real-time capabilities to ensure that the robot control is not affected by system latencies. Furthermore, the Robot Control PC has two ethernet cards to ensure a direct connection with the robot's controller and ROS multi-master ecosystem. Apart from the two ethernet cards, these are rather ordinary desktop computers, as the main task is to generate feasible trajectories of an industrial robot, and there are no other processes that require high-performance capabilities. In this case, the robot control units have the following specifications: Intel Core i7-8400K CPU@ 3.70GHz and 16GB RAM.

The processing unit for computer vision-related computations, on the other hand, requires a

graphics processing unit, both for running ZIVID and for grasp point detection. In this case, the computing capabilities can greatly influence the overall cycle times of the pick & place process. The Computer Vision PC has the following specifications: AMD Ryzen 5 3600@ 3.60 GHz, 32GB RAM and NVIDIA GeForce RTX3060Ti and also with Ubuntu 18.04 system and running ROS melodic.

For training and data generation purposes, an HPC is utilized. The total overall EDI HPC CUDA core count is 62208 and the total overall operational memory – is 352 GB. Each HPC node has a network card with two 10 Gbps ports, connected to the 10 Gbps switch (HP 5900AF-48XG-4QSFP+), which provides high-speed data exchange between HPC nodes.

5.1.2. Software components

Robot Operating System

The whole system communication is done using ROS [161] Melodic Morenia, as it is the ROS version compatible with Ubuntu Linux 18.04 configured on the processing units. All the further described packages are also compatible with this ROS version. ROS-I is being used in the demonstrators as a high-level controller in conjunction with a low-level controller provided by industrial robots.

State Machine

The state machine control of the system is done using the SMACC library for ROS. Conceptually, SMACC defines each individual state machine component as an *Orthogonal*, which also includes a *Client*. *Clients* define connections outside of the state machine. In the state machine, transitions between the states are triggered by *Events* [162].

As possible alternatives for controlling the system in such a manner, there exist such ROS packages as SMACH [163] and Behavior Trees [164]. Behaviour trees offer similar functionalities as state machines. With behaviour trees the development of more complex systems, such as, uncertainties, is possible, but such possibilities can also make the regular development process more complex than necessary, especially when debugging. For this reason, it was chosen to make the system using state machines.

As a different option for implementing state machines, the SMACH package was explored. The previously mentioned SMACC package is based on the SMACH package, the only real difference being that SMACC can be developed in C++, but SMACH - is in the Python programming language. Since the overall system is all in C++, the SMACC package was chosen for easier development.

Data acquisition

An important factor for choosing any hardware that will need to work with an open-source framework, such as ROS, is said hardware's support for this framework. In Subsection 5.1.1, sensors such as Intel Realsense and Zivid are mentioned, and both of them also support being used with ROS, by providing special ROS drivers. These drivers include functions such as capturing an image or a video stream, changing capture settings and so on. Even though the drivers provide sensor functionality in the ROS ecosystem, additional data acquisition and transform package has been developed for unified data acquisition from depth sensors and required data transformations such as 3D coordinate computation from depth data.

It is also important to note that, when using additional sensors in such systems, a special calibration process, called hand-eye calibration, is required, to have all the physical system units operate in a combined coordinate space. This requires acquiring data with both the sensor and the actuator at the same time and then calculating coordinate transformations between them more in detail described in Subsection 3.2.1.

MoveIt!

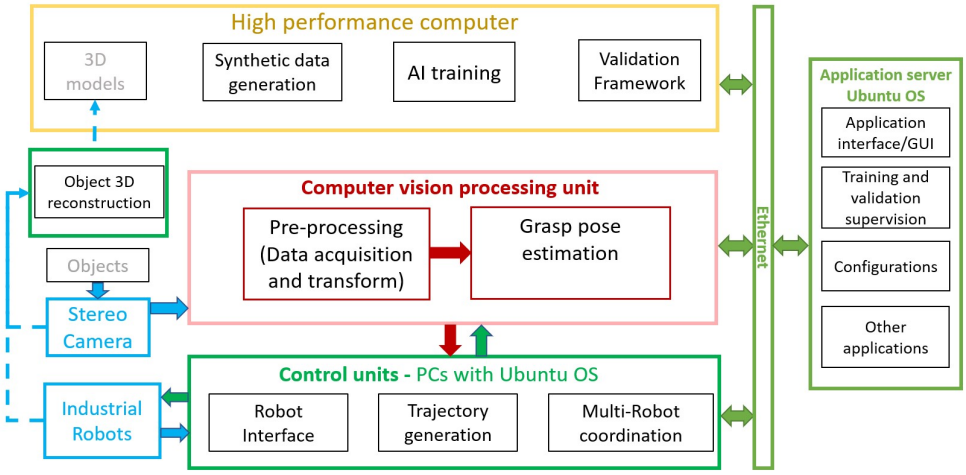
The path planning and execution are done using MoveIt! Motion Planning Framework in detail described in Subsection 3.2.2. In short, it is a collection of tools for robot motion planning, visualization, and execution, which can be used in C++ and Python. A set of sampling-based motion planners are available to choose from for a particular task, however as experimentally confirmed the viability of computed trajectories in Subsection 3.2.2 the STOMP motion planner is used in the demonstrations. MoveIt! also comes with a set of functions to use for robot control which is utilized as well. In addition, it provides the option to visualize a digital twin of the system for easy offline development and can also provide great information about the planned paths in both offline and online scenarios.

Universal Robot driver

Similar to choosing sensors, the choice of the robot itself is also strongly influenced by its support for the framework used. Universal Robot supports work with their robots in the ROS framework by providing the UR ROS driver. This includes all the necessary tools for connecting, communicating, and controlling the robot. Even though the demonstrations are realized with the use of Universal Robots, all the demonstrators are built hardware agnostic, meaning that systems can be adjusted for any ROS-supported industrial robot.

5.2. Multi-robot collaboration for bin-picking task

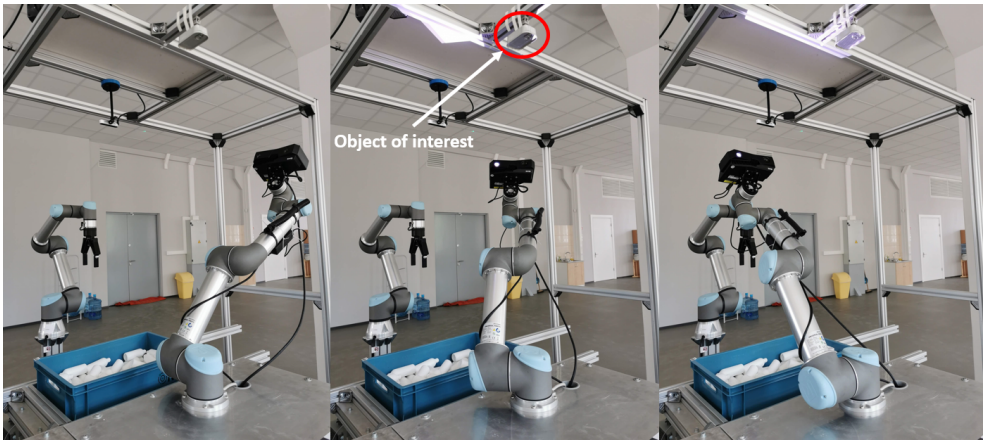
A lot of industrial processes involve operations with a large number of different objects with an arbitrary location. It is hard to automate these kinds of processes because sometimes it is impossible to predetermine the positions of these objects. To overcome this issue, the developed methods within the Thesis are applied in the demonstration of muti-robot collaboration for the bin-picking task. Main functions, including the preparation details for data generation and training, hardware/software components and partitioning are depicted in Fig. 5.4.



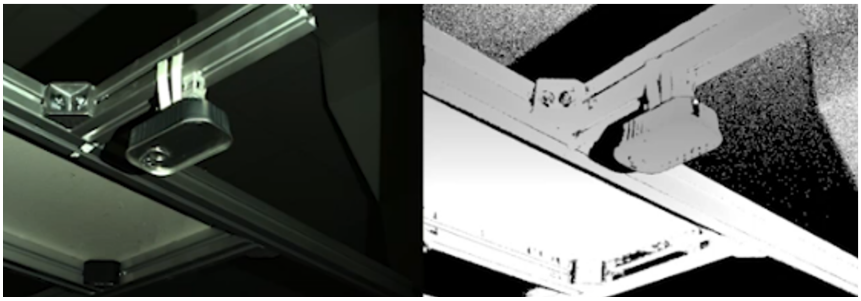
5.4. Fig. Hardware and software partitioning [26].

The synthetic data generation framework described in Section 2.1 is intended to be used with 3D models of the objects of interest, however, in many cases, the models are not available or require manual resources to design and construct them in modelling software. As an example, the bottles used in the synthetic data generator framework as described previously in Section 2.1 were manually designed and constructed using 3D modelling software. Such an approach can lack realistic features which thus can reduce the AI model performance when trained on synthetic data. To cope with this issue, tools for scanning and reconstructing 3D models of the objects of interest were designed and constructed. The scanning software uses a camera mounted on the robot arm, capturing data from different viewpoints, to gather data from all sides of the object of interest – marked with red Fig. 5.5. The point clouds gathered from different viewpoints, where one sample of acquired data is illustrated in Fig. 5.6, are aligned using the camera position information from the robot system. The alignment is fine-tuned by estimating a transformation for point-to-plane distance. After alignment, a 3D mesh is generated representing the object.

Reconstructing an object is done by using a registration algorithm from the open3d library that handles skewed or misalignment information from the point cloud parts. The registration al-



5.5. Fig. Setup and robot movements for object 3D reconstruction [26].



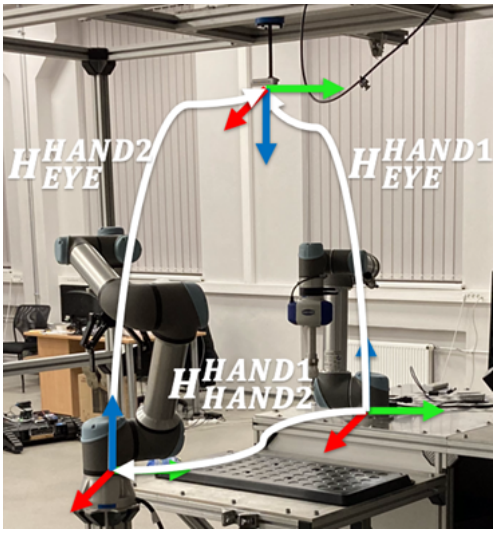
5.6. Fig. Data acquisition for reconstruction purposes [26].

gorithm handles the further post-processing steps and aligns the object point cloud in the proper alignment to construct the object from the gathered point cloud and positioning information. After the point cloud registration process, a point2mesh algorithm is utilized that generates a mesh object from the point cloud object by using a neural network. The scanned objects illustrated in Fig. 5.7 in combination with bottles are then incorporated into the synthetic data generation and training process for the respective computer vision algorithms as described in Chapters 2 and 3 for the realization of computer-vision-based robot control.

For multiple robots to work and interact in the same workspace, they need to be placed in a common coordinate system. Since the system already includes a camera, this common coordinate system can be found using the hand-eye calibration process. This process, where the hand refers to the robot and the eye - to the camera, has already been thoroughly described in the Subsection 3.2.1 for the single robot example. The same principles are used in calibrating the system with two robots. In this scenario, the camera is placed statically, or an EYE-TO-HAND calibration process is being done, with the coordinate systems illustrated in Fig. 5.8.

5.7. Fig. Scene including the plastic bottle- and the reconstructed metal can 3d models (middle) together with the corresponding depth image (left) and segmentation masks (right).

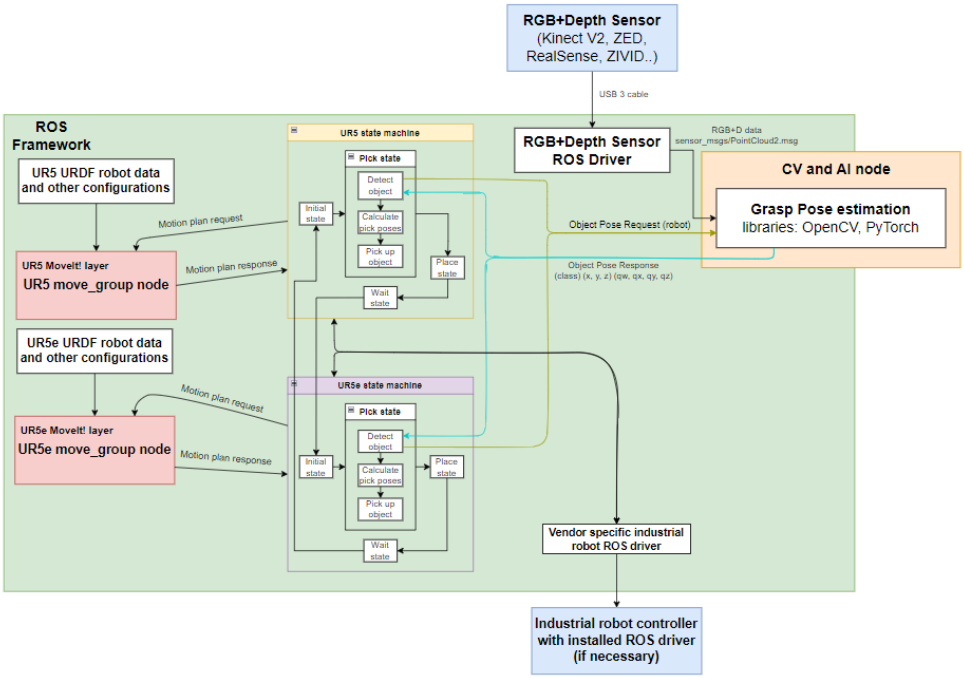
This process is done for each robot separately, and, since the camera is the common object for both robots, its' found location can be used to combine the two robots into one common coordinate system. The process was finding the transformation from the robot base to the camera by calibration, then here it's finding the transformation from the bases of each robot to the camera by two calibrations and using this information to find the third transformation between the bases of each robot. The respective transformation tree is depicted in 1. Appendix.



5.8. Fig. Multi-robot calibration.

Both robots included in the system should be able to do their tasks independently from one another. This raises some issues with the ability to oversee the current states of processes for each element of the system. To improve on these issues and increase modularity, state machines are used. The state machines of both robots are illustrated in Fig. 5.9. There it can be seen that

a single work cycle can be separated into a few states. At first, comes the initial state, where the robot is moved to a safe starting point for further action. Then the following state, at first, provides the robot with an object to be picked up, which is received from the computer vision part. Afterwards, a pickup pose is calculated and in conclusion, if the previous steps have been successful, the object is picked up. Then comes the placing state, where the object is moved to the desired location. And finally, the wait state. At this state, depending on in what way the system is being run, the robot sends a signal to the other robot to start its cycle and waits for the signal back, or it just waits for a signal to start the cycle all over again.



5.9. Fig. Software architecture.

The most crucial robot movements are based on the information given by the computer vision system as described in Subsection 3.1 for rigid objects. It is launched separately from the two robot control programmes. When the robot state machines are launched, they connect to the computer vision system over an IP/TCP connection. When a robot is in a pick state, the first task is detecting the object. This is done by sending a request to the computer vision and artificial intelligence node. All the data is transferred via a standard ROS transport system with request/response semantics. The grasp position (x, y, z) and orientation in quaternion format (qw, qx, qy, qz) as a response to ROS service request. The returned information is an object's grasp position in reference to the camera coordinates. This then needs to be transformed in refer-

ence to the robot base coordinates. At this point, the given object grasp position and orientation are easily understandable by the robot, which means it can safely move on to the next tasks, which are computing the pick poses, motion planning, trajectory generation and execution. The complete computational graph of this demonstration is shown in 2. Appendix.

Overall, the system works reliably and succeeds at the integration of the proposed methods developed within the Thesis. For further illustrative purposes, the demonstration video can be watched by the following link: `https://youtu.be/OezRBPVZ1xU`.

5.3. Handling of waste plastic bottles

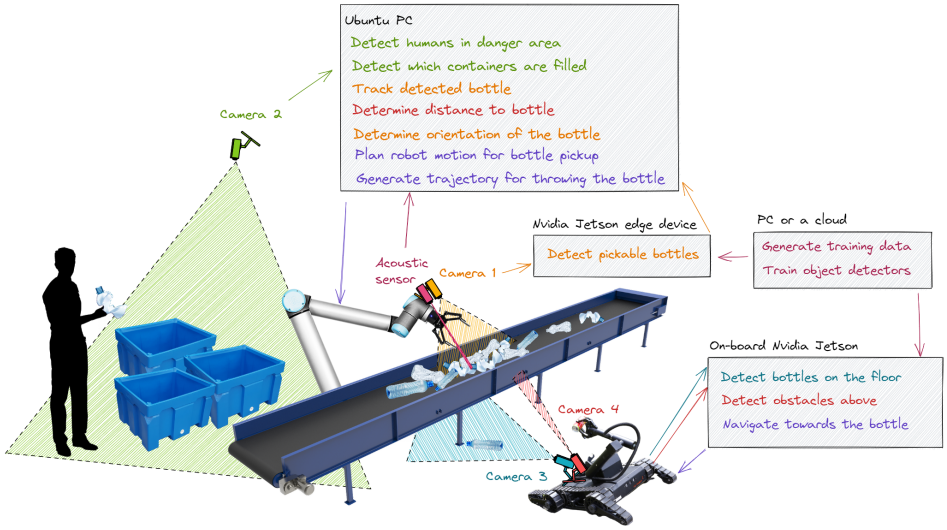
The developed demonstration showcases the use of the proposed methods within the Thesis in an Industry 4.0 application, specifically for automating the sorting of waste plastic bottles. The overall demo is shown in Fig. 5.10 and consists of multiple 3D cameras, an industrial robot arm, and a mobile robot. The main software components are also depicted.

At actual sorting plants, waste bottles are piled on a moving conveyor belt. However, the demo at the laboratory environment is simplified and uses a static table. An Intel RealSense D415 depth camera (Camera 1) and an acoustic sensor are mounted on a robotic arm and directed towards the pile of random and deformed plastic bottles. Sensor data is processed using object detection, tracking, and orientation determination algorithms to plan the motion of the gripper-equipped arm. The algorithm selects an unobstructed bottle, picks it up, and throws it into a container.

Although not addressed by the developments within the Thesis, an extra depth sensor (Camera 2) is positioned above the robot arm and containers. This sensor is utilized to detect empty and full containers, as well as the presence of a human in the robot's working area. If a human is detected, the robot's actions are paused until the human leaves the area.

The demonstration addresses several challenges simultaneously. It is capable of picking objects from piles, even when the objects are identical or highly varied, randomly deformed, and transparent. While the system may be slower than the commonly used jet-of-air type automated sorters, it offers greater versatility and can handle more complex situations and piles. The current gripper-equipped sorters available are only demonstrated in simplified cases, such as picking well-separated objects.

Moreover, this demonstrator aims to achieve more human-like performance by showcasing the throwing functionality typically executed by human workers in manual sorting lines. Although not fully integrated, this demonstration shows the potential of the proposed methods in an Industry 4.0 application.



5.10. Fig. Overview of the demonstrator for sorting waste plastic bottles.

5.3.1. Perception system

The first task of the perception system is to detect those bottles in a random pile that can be picked up by the industrial robot. Most often, those bottles are located on the top of the pile and are unobstructed by other objects. Chosen object detectors are based on deep artificial networks and trained in a supervised fashion. That means a training dataset is needed. It began by acquiring and manually labelling 2030 images of plastic bottle piles. The labelling consisted of drawing bounding boxes around the pickable objects and attaching one of the four class labels to each such object. Besides the point that this is a long and monotonous labour, two main drawbacks of the process were observed:

- Many of the clean plastic bottles are highly transparent. In a pile of such transparent objects, it is often hard for human labellers to precisely indicate the boundaries of pickable bottles.
- The variety of plastic bottles on the market is huge. At sorting plants, those bottles are also randomly deformed and damaged. Therefore, in practical application, there will be a need to create much larger training datasets that include more variations of the bottles. Also, the datasets will have to be occasionally updated by adding examples of new, previously unseen bottles.

With those conclusions, it was decided to develop a data generation module, that would remove the need for manual data labelling, and increase the speed with which the training dataset

can be acquired, labelled, and augmented with new objects as described in Subsection 2.2. The model that achieved the highest precision results was used in this demonstration.

Due to a high amount of variations in object shape and transparency of the object the approach for estimating the grasp pose was different as in the previous demonstrator and steps described for rigid objects in Subsection 3.1. Overall the main task in grasp pose estimation is determining the object's orientation and depth position. As the objects are transparent, the depth readings are rather ambiguous and the object can become invisible to the 3D camera from different viewpoints. Also, due to the deformation of objects and possible holes the grasping of such objects is limited to parallel gripping.

Determining an object's orientation allows for finding an appropriate way to grasp the object. An image processing method for establishing the orientation of an object by using data which is received from the object detector and a depth map is developed. After detecting the bottle information about its locations is acquired, represented as a bounding box drawn in certain coordinates (x1, x2, y1, y2) - the ROI. Inside the ROI, the detected bottle is separated from other bottles and the background. The following pipeline was developed for the processing of the depth map:

- Perform Gaussian Blur to remove noise,
- Apply adaptive threshold to obtain a mask of the foreground,
- Use morphological transformations to fill some gaps in the image,
- Finding contours/shapes of objects,
- Getting minimal object area,
- Calculate the height, width, centre and angle of this minimal area.

5.11. Fig. Full depth map and ROI after thresholding and calculating the angle.

Using the adaptive threshold objects can be separated in different situations. For example, if 1 layer of bottles is lying on the table, then the difference between the bottle and the table will be calculated. If 2 layers of bottles are present, where the first lies on the second, and the second on the table, then the difference between the top layer of bottles and the bottom will be calculated

first. This allows not to change the depth camera settings for different situations. To successfully pick an object one needs to know the angle of yaw rotation (around the Z axis). The orientation of an object is calculated as the angle between the horizontal vector (0°) and the long side of the object. The x-axis will always be horizontal regardless of the camera rotation because it is constant for the longest side of the frame. Determining the minimal area bounding box allows for calculating the width of the detected object. The width of the object is required to determine how wide the robot needs to open the gripper. The minimum area is calculated as pixel values and is converted to real-world centimetres using previously done calibration.

5.12. Fig. A bounding box from object detector and determined angle and width from the proposed approach.

5.3.2. Grasping process

To grasp a detected object, the industrial robot needs to get precise 3D information. Unfortunately, transparent bottles are sometimes also transparent for depth cameras. To tackle this challenge for grasp planning, a narrow beam acoustic sensor was attached to the robotic gripper. Fig. 5.13 depicts two sensors on a robot – a depth camera and an acoustic sensor. By tracking the detected object while moving the industrial robot, it can be placed directly above the bottle. In such a position, the acoustic sensor aims directly under the gripper, giving reliable information about the distance to the object beneath, even if the depth camera doesn't provide complete information about the object of interest.

Even though the grasping process includes additional functionality to grasp transparent plastic bottles, by utilizing the object tracking and acoustic sensor, the construction of robot motions was done by MoveIt! and the state machine similar as in the Section 5.2. The video of the demonstration can be watched by following the link: `https://youtu.be/UFkkfvVdrpI`

5.13. Fig. Sensors attached to the robotic arm.

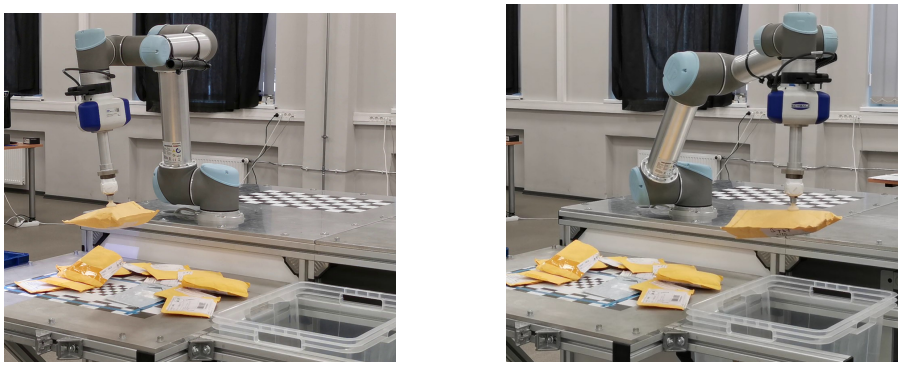
5.4. Post-Office parcel sorting

Internet shopping is becoming more and more popular leading to a rapid increase of postal parcels that need to be processed and shipped. High throughput parcel sorting lines have been developed and utilized in logistic centres that automate most of the work to redistribute the incoming parcels to their destinations. But still, manual work is required to singulate the incoming parcels from the pile and place them on the conveyor belt, forming a bottleneck in otherwise highly elaborate sorting centres.

This seemingly easy and dull job of taking parcels from the pile and placing them on the belt poses several challenges that make its automation difficult. At first, packages come in various sizes, shapes, and colours, so any computer vision algorithm, if to be used, must be highly flexible and not tied to any particular object type. Second, the mix of soft and hard objects, some being very thin, poses challenges to the mechanical design of their transfer mechanisms. Third, objects are piled and partially occluded, so the order in which they can be picked must be taken into account. Fourth, the resulting system has to provide high throughput and work reliably without the need for manual intervention.

In this section, the implemented demonstration that automates the task of post-office parcel pick and place operation is described. It employs Universal Robot UR5 to move parcels from an unstructured pile to the required location. AI-based grasp detection provides high precision of the grasp point. The solution can handle arbitrarily shaped objects, such as postal parcels, without additional training. It easily integrates within any workflow under the ROS framework and successfully addresses the mentioned challenges. In this demonstration, **the main focus is to show the reusability of the developed robot control modules and the viability for the**

integration of different sensory data processing packages.



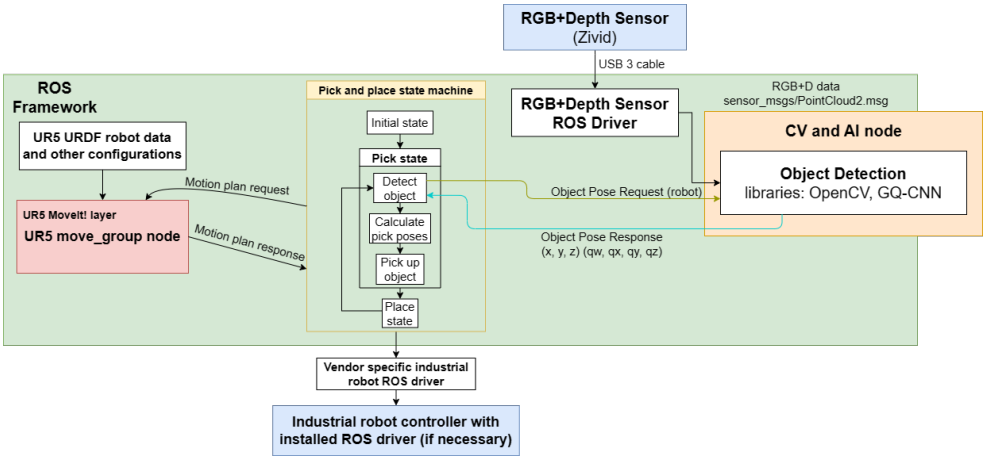
5.14. Fig. Universal Robot UR5 moving around post-office packages.

The proposed robotic system for post office package handling is illustrated in Fig. 5.15 and consists of multiple modules. The whole system is based on the ROS framework, which is used for communicating between modules and controlling the system. The system includes a 6-DOF UR5 robot arm with a vacuum gripper, an RGB-Depth sensor (Zivid), and two processing units for computer-vision-based and robotic motion-based computations, as in detail described in Subsection 5.1.1. The workflow of the system can be described as follows: first, upon request, a 3D image of the workspace is captured with the RGB-Depth sensor and sent to a Grasp Quality Convolutional Neural Network (GQ-CNN) Dex-Net 4.0 [165] for finding a grasp pose.

Second, when a pose is found, it is sent to a robot control state machine in a format of X, Y, and Z coordinates and a W, X, Y, Z quaternion orientation. The robot control state machine is a set of states that describe the whole process in an overseeable and defined way with predictable outcomes. It consists of multiple robot functions such as moving to a desired pose or picking and releasing a desired object. It is implemented using the State Machine Asynchronous C++ (SMACC) [162] package.

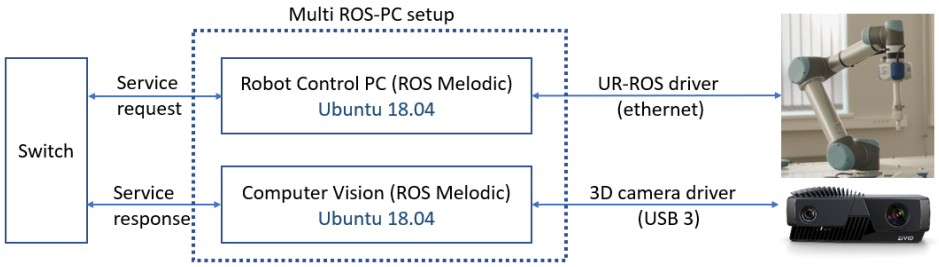
Third, after receiving a pose, a pick path is planned and, if successful, the state machine goes on through a full pick and place cycle. The robot control is implemented using the MoveIt! Motion Planning Framework [166], which manages the planning of paths and communication with the hardware. From the hardware side, the communication is processed using a robot-specific ROS driver by ROS-I in conjunction with a low-level controller provided by the industrial robot. After the object is placed in a predefined location the process starts anew.

The system has been built and tested with the Universal Robots UR5 6-DOF robot arm with an attached Schmalz vacuum gripper for gripping the objects. After examining initial experiments and the nature of computer vision techniques required to handle post office packages, it was determined that the system relies on high precision of the depth information. For this reason, the Zivid camera was chosen to be used in this robotic system. As depicted in Fig. 5.16, the



5.15. Fig. Architecture [22].

proposed system is composed of two processing units described in detail in Subsection 5.1.1, which are configured as a Multi ROS-PC setup.



5.16. Fig. Hardware setup [22].

5.4.1. Communication and synchronization

Since the whole system is controlled by the state machine, the system functions by transitioning from one definite state to the next. As depicted in Fig. 5.16, the robot control and workspace sensing are being processed on two different units. This provides an extra layer of complexity to the system in the form of communication and synchronization.

The signal to start detection can be sent once the robot is no longer blocking the workspace, but this happens multiple states before the robot is ready to begin a new cycle. Since the grasp point detection takes an unpredictable amount of time, the robot system can both run forward or fall behind during the process in terms of synchronizing when the new pick point is sent. To make

the process more time efficient, the communication between processing units happens in parallel with the robot's motion. When a new grasp point is detected, the point is not sent as a message, but rather, it is set as a new value on the ROS parameter server with the tag *new*, describing that the robot hasn't yet used this point. Once the robot has reached a state where it can act upon a new point, it checks this *new* tag, which can be set to either true or false. Depending on when both of these parts of the system reach this point, the robot either waits until a new pose with this tag set to true appears or reaches the new cycle starting pose and instantly starts working on the new cycle. In this way, the two parts of the system are both co- and independent in their functionality.

5.4.2. Grasp point detection

As the postal parcels have irregular shapes, see Fig. 5.17(**a**), for grasp point calculation, the Dex-Net 4.0 [165] library has been selected, which delivers robust performance in our application. It has been configured to utilize the Grasp Quality Convolutional Neural Networks (GQ-CNNs) method for grasp detection, and it delivers information for a system with a suction gripper, for which it is well suited. The camera intrinsics are obtained directly from the camera using its ROS driver, and the calibration is done beforehand. The chosen Zivid camera naturally produces images in 1920x1200 resolution, and such images take about 10 seconds to process with Dex-Net. To increase the performance the images are, firstly, cropped to show only the region of interest (ROI) where the packages reside. Afterwards, they are downscaled to 640x480.

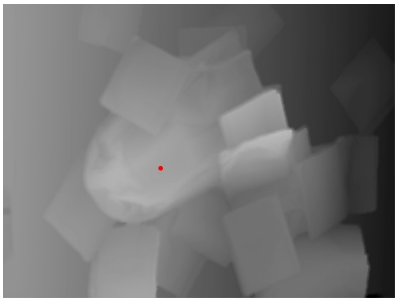
The output quality of the Dex-Net is greatly enhanced if one provides a segmentation mask in its input showing precisely the region occupied with objects (the objects do not need to be segmented individually). To produce such a mask, the depth image of a clean table is taken at the calibration phase, and when operating, calculate the segmentation mask showing only pixels with a depth of fewer than 2 millimetres (minimum parcel height) above the calibrated table. Such a setup allows Dex-Net to produce reliable grasps, see Fig. 5.17.

5.4.3. Robot control

The control system consists of multiple states that manage different robot actions. The whole robot movement process is divided into several states, such as moving to a predefined start position, moving to an ever-changing pick position, activating the gripper, and so on.

Trajectory planning for movement in these states is done using MoveIt!, which is connected to the robot and gripper drivers with the move group interface. In the MoveIt! setup of the system, the planners best suited for the job can be chosen from the ones provided in the OMPL. The physical setup of the system is also represented digitally in the MoveIt! setup. Both the

(**a**) A pile of postal packages



(**b**) The detected grasp point

5.17. Fig. Dex-Net input and output. It takes an RGBD image of the object pile as input and produces a grasp point.

table and the robot has a 3D model, which represents the system in real time. This provides the necessary information for the planner to plan safe, collision-free trajectories. This also allows the operator of the system to see both the planned trajectories and found object coordinates before they are acted upon.

5.4.4. Results and discussion

The described system was implemented in the lab environment and validated for post office parcel handling in multiple scenarios, illustrated in Fig. 5.18. The experiments were done in 5 different scenarios, each consisting of a different amount of packages, and the system was tested for its ability to clear the workspace. The physical setup of the robot is depicted in Fig. 5.14. The robot is located near the table holding a pile of realistic parcels, see Fig. 5.17(**a**), and they should be moved to a container standing nearby. The camera is mounted about 1m above the table.

5.18. Fig. Testing scenario examples. From left to right, 1, 5, 10, 15, 20 packages in the workspace.

For testing, five scenarios, consisting of 1, 5, 10, 15 and 20 parcels in the workspace, were made. Each scenario was tested ten times, whereas the packages were randomly distributed in the workplace. The packages were piled on top of each other when possible to replicate real-

life setups as much as possible. The attempts for each scenario were counted and compared to the expected amount, which was set to one attempt per package in the scenario. Afterwards, the average deviation from the expected attempt amount was calculated, and lastly, an overall average deviation was calculated to see the mean error of the system. The results can be seen in Table 5.1.

5.1. table

Experimental results [22].

Scenario, packages	Grasp attempts, average	Additional grasps	Success rate
1	1	0.00 %	100 %
5	5.3	5.66 %	100 %
10	10.4	3.85 %	100 %
15	16	6.25 %	100 %
20	21.3	6.10 %	100 %
	Overall additional grasps	4.37 %	

As per results, the one package scenario showed a perfect score, five packages resulted in an average of 5.3 attempts per 5 packages, for 10 packages it took an average of 10.4 attempts, for 15 packages an average of 16 attempts were needed, and for 20 packages an average of 21.3 attempts was required. Overall the system achieved a **100 %** success rate for emptying the workspace, meaning that all the packages have been successfully handled, however, an average of 4.37 % increase in additional grasps can be observed when compared to the perfect 1 attempt per package. By measuring only one attempt per package, the system succeeded with **94.1 %** successful grasps. Even though some grasps didn't succeed in a success, the system could recover and grasp the object on the second try.

While testing, it was observed that most of the failed attempts can be attributed to a grasp pose located on an uneven part where the vacuum gripper cannot achieve the required vacuum to grasp the desired package. This issue could be addressed by a more flexible vacuum cap or increasing the vacuum, whereas the available hardware in this experimental setup was already adjusted as much as possible for these experiments. A customized vacuum system for specific use cases and package materials could increase the average grasping success rates. In a real use case, there would be a conveyor belt in the place of the container. The system has been tested with different pile configurations and parcel types, and in all cases, the system worked reliably, as can be seen in the video demonstration `https://www.youtube.com/watch?v=djxrbQZtiKk`.

5.5. Summary

The implemented demonstrations effectively tackle several challenges within the manufacturing environment, showcasing the practicality and effectiveness of the developed techniques within smart control of industrial robots. The experimental setup and demonstrations show the seamless integration of diverse hardware and software building blocks, highlighting the overall modularity and adaptability. Additionally, the demonstrations are showcasing the addressed characteristics of maintainability and usability.

The first demonstration presents a powerful solution for automating industrial processes involving a large number of simple-shaped objects positioned arbitrarily and overlapping each other in a pile. By utilizing computer-vision-based robot control the system accurately identifies, localizes and manipulates objects in the workspace. The integration of synthetic data generation enhances the performance of AI models for precise grasping and manipulation. Hand-eye calibration, however, ensures seamless collaboration between multiple robots, enabling them to work independently within a common coordinate system. The state machine architecture enhances modularity and oversight of the system's state during task execution. Overall, the demonstrated system showcases a reliable and efficient approach to automation, paving the way for more streamlined and effective industrial processes with variable object locations.

In a similar manner, the second demonstration deploys computer-vision-based robot control driven by synthetic data, but for grasping transparent plastic bottles. By processing sensor data through object detection, tracking, and orientation determination algorithms, the system plans the robot arm's motion to pick up unobstructed bottles and throw them into containers. This demonstration successfully addresses challenges associated with picking deformed and transparent plastic bottles which are overlapping each other in a pile, highlighting the potential of the proposed methods in industry applications.

The implemented system for post-office parcel pick and place operation emphasizes the modularity of the developed robot control approaches. The system efficiently moves parcels from an unstructured pile to a designated location. The solution employs AI-based grasp detection through the Dex-Net 4.0 library, enabling it to handle parcels of varying sizes, shapes, and colours without additional training. The system's performance is showcased and evaluated, proving its potential for efficient and reliable post-office parcel sorting solutions.

6. CONCLUSIONS

the Thesis addresses the smart control methods for industrial robots, their applications, and learning strategies. The primary aim of the Thesis was to research and develop novel techniques in the fields of efficient data preparation methods, computer-vision-based robot control and knowledge acquisition from humans through demonstrations and, therefore, improve the deployment of industrial robots for working in dynamic environments and advancing the operability of such systems.

To achieve the set aim, six tasks were defined and completed.

1. To perform a review of the literature on smart control methods for industrial robots. This task was accomplished in Chapter 1. The literature review acknowledged the current state-of-the-art approaches for the smart control of industrial robots. Different control methods and learning strategies were overviewed in detail, which also led to the identification of current challenges and open issues described in Section 1.5, being a basis for the proposed methods in the Thesis. Even though the methods related to smart control of industrial robots have gone through remarkable developments in the last decade, there still are several challenges and unsolved issues, namely within efficient data preparation methods for computer-vision-based robot control and knowledge acquisition process.

2. To formalize the concept of smart control of industrial robots. Which was accomplished during the extensive literature review in Chapter 1. The identification of core functionalities within the smart control of industrial robots was fueled by the exploration of current trends in robot control, leading to a formalization of the concept of smart control of industrial robots following the principle of "see-think-act" as illustrated in Fig. 1.1. The concept is a broad view of smart industrial robots consisting of such functionalities as perception; high-level instructions and context-aware task execution; knowledge acquisition and generalization; adaptive planning. Accordingly, not all of the functionalities are addressed within the Thesis, but the ones that were recognized with the highest potential to address the identified challenges and open issues in Section 1.5.

3. To research and develop efficient data preparation methods for computer-vision-based robot control. This task was accomplished in Chapter 2 by developing a methodology for synthetic data generation and further built on in Chapter 3 accomplishing a computer-vision-based robot control. Synthetic data generation frameworks show promising results in deep learning-based object detection tasks and can not only supplement real data when the variety of real training data is insufficient but also completely replace real data and still perform reliably in real-world settings. The use of synthetic data generation in robotic systems remarkably decreases the required time and manual efforts within the data generation process. Thus the versatile data that can be synthetically generated can be used for validation purposes, therefore evaluating the system's performance before it is deployed. The tentative results can show the viability of the proposed algorithms in the specific use cases, and if the achieved results are not

satisfactory, the conclusion of the need for improvements or additional training can be achieved without running the system in real life.

4. To research and develop a methodology for human demonstration incorporation in the knowledge acquisition process. The human demonstration incorporation in the knowledge acquisition process, namely imitation learning-based control, was accomplished in Chapter 4. The original goal set out to accomplish was to devise a framework for recording demonstrations by human actors, and using these to execute an object-throwing task as a stand-in for a variety of similar future applications was achieved. By using motion capture as the data collection mechanism, it proved possible to employ a series of analytic pre-processing steps to turn raw recording data into demonstration data sets suitable for the knowledge acquisition process for industrial robots. The data collection step with motion capture equipment proved to be straightforward to customize for the throwing task. However, it did require some attention to how the physical demonstrations were performed – with allowances for automatic demarcation of distinct demonstrations. A pre-processing pipeline had to be developed to extract these separate trajectories and only their relevant segments, as well as infer hidden state variables through indirect observation. While the latter will likely be specific to each application, the methods herein are adaptable for different scenarios.

5. To apply proposed methods in demonstrations. In total, the proposed methods were applied in three demonstrations described in Chapter 5 addressing current challenges in various robotic grasping applications. In the first demonstrator: "Multi-robot collaboration for the bin-picking task", the synthetic data generation and grasp pose estimation methods proposed in Section 2.1 and Chapter 3 have been successfully applied and demonstrated for reliable grasping and sorting of two different objects mixed in a pile. The second demonstrator, "Handling of waste plastic bottles", however, applied the data generation described in Section 2.2 for computer-vision-based robot control and imitation-learning-based robotic object throwing learned from human demonstrations described in Chapter 4. Last but not least, the demonstrator: "Post-Office parcel sorting", shows the modularity and the reusability of the developed robot control modules and the viability of the integration of different sensory data processing packages.

6. To draw conclusions about the results of the Thesis. The final task of the Thesis is done in the current Chapter. The main conclusion about smart control methods for industrial robots developed in the Thesis is that their properties and test results demonstrate that the aim of the Thesis is successfully achieved – the developed methods improve the deployment of industrial robots for working in dynamic environments and advancing the operability of such systems. In addition, the accomplished tasks prove three statements defined at the beginning of the Thesis.

Statement 1. The utilization of synthetic data in the training process for simple-shaped object detection in bin-picking setup reduces the required manual effort, whilst trained models are able to detect at least one object in every scene with an IoU threshold above 0.95. The proposed synthetic data generation methodology for training data acquisition in Chapter

2 demonstrates its feasibility and potential for industrial robot pick and place tasks in a bin-picking setup. The real data still outperforms synthetic data if solely the object detectors' average precision is analysed, however, in smart control of industrial robots, it is not that crucial to have precise information about all the objects in the scene, as typically only one object can be grasped anyway, after which the scene has changed and needs to be perceived again. For this aspect, another precision metric was introduced. Namely, it was measured in how many scenes at least one object with an IoU threshold above 0.95 can be detected. The achieved results in Section 2.1.4 prove the first statement.

Statement 2. For computer-vision-based robot control in the case of simple-shaped objects, at least one valid grasp can be found in bin-picking setup in more than 99 % of scenes when computer-vision algorithms have been trained solely on synthetic data. By exploiting the synthetic data generation methodology, the respective deep-learning models in computer-vision-based robot control for grasp pose estimation are trained solely on synthetic data. The synthetic image generator also takes on a critical role in the validation process. Its versatility allows for the rapid evaluation of various aspects of the proposed algorithms, providing tentative precision results that demonstrate the viability of a specific use case. The achieved results in Section 3.3 prove the second statement.

Statement 3. The learned trajectories for an object-throwing task by utilizing indirect human demonstrations achieve cosine similarity of more than 0.98 when compared to the validation dataset and, when executed on a real robot, closely resemble the expert human throwing motions. Human demonstrations are recorded with motion capture equipment, a split and cropped data set is augmented with target coordinates, and a gripper actuation signal is created. This is used to train neural network models in behavioural cloning. When operating in the open-loop regime, prior model outputs are used to autoregressively predict subsequent states. The resulting sequence can then be fed into a Cartesian motion planner to control a robot. The neural networks were trained with the main goal of generating the position and orientation time series of the gripper and replicating the expert behaviour as closely as possible. The achieved results in Section 4.4 prove the third statement.

Bibliography

[1] J. Wallén, *The history of the industrial robot*. Linköping University Electronic Press, 2008.

[2] "Iso 10218-2:2011. robots and robotic devices — safety requirements for industrial robots — part 2: Robot systems and integration," 2011.

[3] M. Wilson, "Chapter 2 - industrial robots," in *Implementation of Robot Systems* (M. Wilson, ed.), pp. 19–38, Oxford: Butterworth-Heinemann, 2015.

[4] J. Carlsson, *A decade of robotics:[analysis of the diffusion of industrial robots in the 1980s by countries, application areas, industrial branches and types of robots]*. Sveriges mekanförb., 1991.

[5] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, "Intelligent manufacturing in the context of industry 4.0: A review," *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[6] "Executive summary world robotics 2021 industrial robots," 2022.

[7] L. Sanneman, C. Fourie, and J. A. Shah, "The state of industrial robotics: Emerging technologies, challenges, and key research directions," *arXiv preprint arXiv:2010.14537*, 2020.

[8] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, 2012.

[9] L. Evjemo, T. Gjerstad, E. Grøtli, and G. Sziebig, "Trends in smart manufacturing: Role of humans and industrial robots in smart factories," *Current Robotics Reports*, vol. 1, 04 2020.

[10] L. Probst, B. Pedersen, V. Lefebvre, and L. Dakkak, "Usa-china-eu plans for ai: where do we stand," *Digital Transformation Monitor of the European Commission*, 2018.

[11] J. Arents, V. Abolins, J. Judvaitis, O. Vismanis, A. Oraby, and K. Ozols, "Human–robot collaboration trends and safety aspects: A systematic review," *Journal of Sensor and Actuator Networks*, vol. 10, no. 3, 2021.

[12] P. Osterrieder, L. Budde, and T. Friedli, "The smart factory as a key construct of industry 4.0: A systematic literature review," *International Journal of Production Economics*, vol. 221, p. 107476, 2020.

[13] P. K. R. Maddikunta, Q.-V. Pham, B. Prabadevi, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: a survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, p. 100257, 2021.

[14] W. Golnazarian and E. Hall, "Intelligent industrial robots," 09 2002.

[15] L. Shao and R. Volz, "Methods and strategies of object localization," 1989.

[16] C. Premebida, R. Ambrus, and Z.-C. Marton, "Intelligent robotic perception systems," 2018.

[17] J. Arents, R. Cacurs, and M. Greitans, "Integration of computervision and artificial intelligence subsystems with robot operating system based motion planning for industrial robots," *Automatic Control and Computer Sciences*, vol. 52, no. 5, pp. 392–401, 2018.

[18] J. Arents and M. Greitans, "Smart industrial robot control trends, challenges and opportunities within manufacturing," *Applied Sciences*, vol. 12, no. 2, p. 937, 2022.

[19] P. Torres, J. Arents, H. Marques, and P. Marques, "Bin-picking solution for randomly placed automotive connectors based on machine learning techniques," *Electronics*, vol. 11, no. 3, 2022.

[20] P. Racinskis, J. Arents, and M. Greitans, "A motion capture and imitation learning based approach to robot control," *Applied Sciences*, vol. 12, no. 14, 2022.

[21] V. Feščenko, J. Ārents, and R. Kadiķis, "Synthetic data generation for visual detection of flattened pet bottles," *Machine Learning and Knowledge Extraction*, vol. 5, no. 1, pp. 14–28, 2023.

[22] O. Vismanis, J. Arents, K. Freivalds, V. Ahluwalia, and K. Ozols, "Robotic system for post office package handling," *Applied Sciences*, vol. 13, no. 13, 2023.

[23] E. Buls, R. Kadikis, R. Cacurs, and J. Ārents, "Generation of synthetic training data for object detection in piles," in *Eleventh International Conference on Machine Vision (ICMV 2018)* (A. Verikas, D. P. Nikolaev, P. Radeva, and J. Zhou, eds.), vol. 11041, p. 110411Z, International Society for Optics and Photonics, SPIE, 2019.

[24] J. Arents, B. Lesser, A. Bizuns, R. Kadikis, E. Buls, and M. Greitans, "Synthetic data of randomly piled, similar objects for deep learning-based object detection," in *Image Analysis and Processing – ICIAP 2022*, (Cham), pp. 706–717, Springer International Publishing, 2022.

[25] D. Duplevska, M. Ivanovs, J. Arents, and R. Kadikis, "Sim2real image translation to improve a synthetic dataset for a bin picking task," in *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–7, IEEE, 2022.

[26] J. Arents, M. Greitans, and B. Lesser, *Construction of a Smart Vision-Guided Robot System for Manipulation in a Dynamic Environment*, pp. 205–220. 09 2021.

[27] G. Urlini, J. Arents, and A. Latella, *AI in Industrial Machinery*, pp. 179–185. 09 2021.

[28] P. Li and X. Liu, "Common sensors in industrial robots: A review," vol. 1267, p. 012036, jul 2019.

[29] S. Mittal, M. A. Khan, D. Romero, and T. Wuest, "Smart manufacturing: Characteristics, technologies and enabling factors," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 233, no. 5, pp. 1342–1361, 2019.

[30] M. Abubakr, A. T. Abbas, I. Tomaz, M. S. Soliman, M. Luqman, and H. Hegab, "Sustainable and smart manufacturing: An integrated approach," *Sustainability*, vol. 12, no. 6, 2020.

[31] K. Wei and B. Ren, "A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved rrt algorithm," *Sensors*, vol. 18, no. 2, p. 571, 2018.

[32] D. Vernon and M. Vincze, "Industrial priorities for cognitive robotics.," in *EUCognition*, pp. 6–9, 2016.

[33] G. Kraetzschmar, "Software engineering factors for cognitive robotics," 2018.

[34] H. Samani, *Cognitive robotics*. CRC Press, 2015.

[35] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[36] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

[37] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, 2019.

[38] V. Kakani, V. H. Nguyen, B. P. Kumar, H. Kim, and V. R. Pasupuleti, "A critical review on computer vision and artificial intelligence in food industry," *Journal of Agriculture and Food Research*, vol. 2, p. 100033, 2020.

[39] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," 06 2013.

[40] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," vol. 2017-January, pp. 6517–6525, 2017. cited By 3103.

[41] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," vol. 2015-January, pp. 91–99, 2015. cited By 10739.

[42] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019.

[43] C. Poss, O. B. Mlouka, T. Irrenhauser, M. Prueglmeier, D. Goehring, F. Zoghlami, and V. Salehi, "Robust framework for intelligent gripping point detection," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 717–723, 2019.

[44] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust robot vacuum suction grasp targets in point clouds using a new analytic model and deep learning," 2018.

[45] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, 2019.

[46] W. Lin, A. Anwar, Z. Li, M. Tong, J. Qiu, and H. Gao, "Recognition and pose estimation of auto parts for an autonomous spray painting robot," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1709–1719, 2019.

[47] L. Zhang, Q. Ye, W. Yang, and J. Jiao, "Weld line detection and tracking via spatial-temporal cascaded hidden markov models and cross structured light," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 4, pp. 742–753, 2014.

[48] A. Khan, C. Mineo, G. Dobie, C. Macleod, and G. Pierce, "Vision guided robotic inspection for parts in manufacturing and remanufacturing industry," *Journal of Remanufacturing*, vol. 11, no. 1, pp. 49–70, 2021.

[49] O. Skotheim, M. Lind, P. Ystgaard, and S. A. Fjerdingen, "A flexible 3d object localization system for industrial part handling," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3326–3333, 2012.

[50] M. Tsai, J.-J. Fang, and J.-L. Chang, "Robotic path planning for an automatic mold polishing system," *International Journal of Robotics & Automation - INT J ROBOTICS AUTOM*, vol. 19, 01 2004.

[51] X. Zhen, J. C. Y. Seng, and N. Somani, "Adaptive automatic robot tool path generation based on point cloud projection algorithm," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 341–347, 2019.

[52] R. Peng, D. Navarro-Alarcon, V. Wu, and W. Yang, "A point cloud-based method for automatic groove detection and trajectory generation of robotic arc welding tasks," 04 2020.

[53] M. Fujita, Y. Domae, A. Noda, G. Garcia Ricardez, T. Nagatani, A. Zeng, S. Song, A. Rodriguez, A. Causo, I.-M. Chen, *et al.*, "What are the important technologies for bin picking? technology analysis of robots in competitions based on a set of performance metrics," *Advanced Robotics*, vol. 34, no. 7-8, pp. 560–574, 2020.

[54] B. Horn and K. Ikeuchi, "The mechanical manipulation of randomly oriented parts," *Scientific American - SCI AMER*, vol. 251, pp. 100–111, 08 1984.

[55] J. A. Marvel, K. Saidi, R. Eastman, T. Hong, G. Cheok, and E. Messina, "Technology readiness levels for randomized bin picking," in *Proceedings of the Workshop on Performance Metrics for Intelligent Systems*, pp. 109–113, 2012.

[56] D. Holz, A. Topalidou-Kyniazopoulou, J. Stückler, and S. Behnke, "Real-time object detection, localization and verification for fast robotic depalletizing," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1459–1466, 2015.

[57] K. Kleeberger, R. Bormann, W. Kraus, and M. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, vol. 1, p. 239–249, 12 2020.

[58] F. Spenrath and A. Pott, "Gripping point determination for bin picking using heuristic search," *Procedia CIRP*, vol. 62, pp. 606–611, 12 2017.

[59] R. He, J. Rojas, and Y. Guan, "A 3d object detection and pose estimation pipeline using rgb-d images," 2017.

[60] J. Sock, K. Kim, C. Sahin, and T.-K. Kim, "Multi-task deep networks for depth-based 6d object pose and joint registration in crowd scenarios," 2019. cited By 6.

[61] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," 2017.

[62] A. Olesen, B. Gergaly, E. Ryberg, M. Thomsen, and D. Chrysostomou, "A collaborative robot cell for random bin-picking based on deep learning policies and a multi-gripper switching strategy," vol. 51, pp. 3–10, 2020. cited By 0.

[63] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," 2018.

[64] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 2018.

[65] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," 2018.

[66] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, 02 2009.

[67] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," 2018.

[68] F. Zoghlami, P. Kurrek, M. Jocas, G. Masala, and V. Salehi, "Design of a deep post gripping perception framework for industrial robots," *J. Comput. Inf. Sci. Eng.*, vol. 21, 2021.

[69] R. Matsumura, Y. Domae, W. Wan, and K. Harada, "Learning based robotic bin-picking for potentially tangled objects," pp. 7990–7997, 11 2019.

[70] M. Moosmann, F. Spenrath, K. Kleeberger, M. U. Khalid, M. Mönnig, J. Rosport, and R. Bormann, "Increasing the robustness of random bin picking by avoiding grasps of entangled workpieces," *Procedia CIRP*, vol. 93, pp. 1212–1217, 2020. 53rd CIRP Conference on Manufacturing Systems 2020.

[71] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 01 2016.

[72] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

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

[74] R. Liu, F. Nageotte, P. Zanne, M. de Mathelin, and B. Dresp-Langley, "Deep reinforcement learning for the control of robotic manipulation: A focussed mini-review," *Robotics*, vol. 10, no. 1, 2021.

[75] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms.," *J. Mach. Learn. Res.*, vol. 22, pp. 30–1, 2021.

[76] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *International conference on machine learning*, pp. 703–711, PMLR, 2017.

[77] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[78] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning," *Sensors*, vol. 21, no. 4, p. 1278, 2021.

[79] A. Zhan, P. Zhao, L. Pinto, P. Abbeel, and M. Laskin, "A framework for efficient robotic manipulation," 2020.

[80] D. Quillen, E. Jang, O. Nachum, C. Finn, J. Ibarz, and S. Levine, "Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6284–6291, IEEE, 2018.

[81] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 1461–1466, 2020.

[82] Y. Wang, X. Lan, C. Feng, L. Wan, J. Li, Y. Liu, and D. Li, "An experience-based policy gradient method for smooth manipulation," in *2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pp. 93–97, 2019.

[83] O.-M. Pedersen, E. Misimi, and F. Chaumette, "Grasping unknown objects by coupling deep reinforcement learning, generative adversarial networks, and visual servoing," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5655–5662, 2020.

[84] Y. Chen, Z. Ju, and C. Yang, "Combining reinforcement learning and rule-based method to manipulate objects in clutter," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, IEEE, 2020.

[85] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," pp. 4238–4245, 10 2018.

[86] Y. Yang, H. Liang, and C. Choi, "A deep learning approach to grasping the invisible," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2232–2239, 2020.

[87] K. Shin, M. Sim, E. Choi, H. Park, J.-W. Choi, Y. Cho, J. I. Sohn, S. N. Cha, and J. E. Jang, "Artificial tactile sensor structure for surface topography through sliding," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 6, pp. 2638–2649, 2018.

[88] N. Vulin, S. Christen, S. Stevšić, and O. Hilliges, "Improved learning of robot manipulation tasks via tactile intrinsic motivation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2194–2201, 2021.

[89] H. Merzić, M. Bogdanović, D. Kappler, L. Righetti, and J. Bohg, "Leveraging contact forces for learning to grasp," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3615–3621, IEEE, 2019.

[90] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, IEEE, 2019.

[91] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, "Learning force control for contact-rich manipulation tasks with rigid position-controlled robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5709–5716, 2020.

[92] S. H. Huang, M. Zambelli, J. Kay, M. F. Martins, Y. Tassa, P. M. Pilarski, and R. Hadsell, "Learning gentle object manipulation with curiosity-driven deep reinforcement learning," *arXiv preprint arXiv:1903.08542*, 2019.

[93] A. Melnik, L. Lach, M. Plappert, T. Korthals, R. Haschke, and H. Ritter, "Tactile sensing and deep reinforcement learning for in-hand manipulation tasks," in *IROS Workshop on Autonomous Object Manipulation*, 2019.

[94] A. Hundt, B. Killeen, N. Greene, H. Wu, H. Kwon, C. Paxton, and G. D. Hager, ""good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6724–6731, 2020.

[95] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from cad," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3524–3531, IEEE, 2018.

[96] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement learning on variable impedance controller for high-precision robotic assembly," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3080–3087, 2019.

[97] P. Chen and W. Lu, "Deep reinforcement learning based moving object grasping," *Information Sciences*, vol. 565, pp. 62–76, 2021.

[98] Z. Hu, Y. Zheng, and J. Pan, "Living object grasping using two-stage graph reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1950–1957, 2021.

[99] A. Attia and S. Dayan, "Global overview of imitation learning," *arXiv preprint arXiv:1801.06503*, 2018.

[100] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *arXiv preprint arXiv:1811.06711*, 2018.

[101] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY …, 1989.

[102] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *2009 IEEE International Conference on Robotics and Automation*, pp. 763–768, IEEE, 2009.

[103] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," in *In AISTATS*, Citeseer, 2011.

[104] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 1, 2004.

[105] J. Ho and S. Ermon, "Generative adversarial imitation learning," *Advances in neural information processing systems*, vol. 29, pp. 4565–4573, 2016.

[106] F. Torabi, G. Warnell, and P. Stone, "Generative adversarial imitation from observation," *arXiv preprint arXiv:1807.06158*, 2018.

[107] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *International Journal of Intelligent Robotics and Applications*, vol. 3, no. 4, pp. 362–369, 2019.

[108] M. Suomalainen, F. J. Abu-Dakka, and V. Kyrki, "Imitation learning-based framework for learning 6-d linear compliant motions," *Autonomous Robots*, vol. 45, no. 3, pp. 389–405, 2021.

[109] T. Gašpar, B. Nemec, J. Morimoto, and A. Ude, "Skill learning and action recognition by arc-length dynamic movement primitives," *Robotics and autonomous systems*, vol. 100, pp. 225–235, 2018.

[110] C. A. V. Perico, J. De Schutter, and E. Aertbeliën, "Combining imitation learning with constraint-based task specification and control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1892–1899, 2019.

[111] S. Gubbi, S. Kolathaya, and B. Amrutur, "Imitation learning for high precision peg-in-hole tasks," in *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 368–372, IEEE, 2020.

[112] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635, IEEE, 2018.

[113] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1134–1141, IEEE, 2018.

[114] M. Edmonds, F. Gao, X. Xie, H. Liu, S. Qi, Y. Zhu, B. Rothrock, and S.-C. Zhu, "Feeling the force: Integrating force and pose for fluent discovery through imitation learning to open medicine bottles," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3530–3537, IEEE, 2017.

[115] M. Andtfolk, L. Nyholm, H. Eide, and L. Fagerström, "Humanoid robots in the care of older persons: A scoping review," *Assistive Technology*, pp. 1–9, 2021.

[116] J. S. Dyrstad, E. R. Øye, A. Stahl, and J. R. Mathiassen, "Teaching a robot to grasp real fish by imitation learning from a human supervisor in virtual reality," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7185–7192, IEEE, 2018.

[117] A. Jha, S. S. Chiddarwar, R. Y. Bhute, V. Alakshendra, G. Nikhade, and P. M. Khandekar, "Imitation learning in industrial robots: a kinematics based trajectory generation framework," in *Proceedings of the Advances in Robotics*, pp. 1–6, 2017.

[118] R. Vuga, M. Ogrinc, A. Gams, T. Petrič, N. Sugimoto, A. Ude, and J. Morimoto, "Motion capture and reinforcement learning of dynamically stable humanoid movement primitives," in *2013 IEEE International Conference on Robotics and Automation*, pp. 5284–5290, 2013.

[119] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent network models for human dynamics," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[120] D. Pavllo, D. Grangier, and M. Auli, "Quaternet: A quaternion-based recurrent model for human motion," *arXiv preprint arXiv:1805.06485*, 2018.

[121] J. M. de Vet, D. Nigohosyan, J. N. Ferrer, A.-K. Gross, S. Kuehl, and M. Flickenschild, *Impacts of the COVID-19 pandemic on EU industries*. European Parliament, 2021.

[122] N. Jakobi, P. Husb, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," 02 1999.

[123] "Systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models.." `https://iso25000.com/index.php/en/iso-25000-standards/iso-25010`. Accessed: 2021-04-17.

[124] "Systems and software engineering — systems and software quality requirements and evaluation (square) — measurement of system and software product quality." `https://www.iso.org/standard/35747.html`. Accessed: 2021-04-25.

[125] S. Karnouskos, R. Sinha, P. Leitão, L. Ribeiro, and T. I. Strasser, "Assessing the integration of software agents and industrial automation systems with iso/iec 25010," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pp. 61–66, 2018.

[126] S. Karnouskos, R. Sinha, P. Leitão, L. Ribeiro, and T. I. Strasser, "The applicability of iso/iec 25023 measures to the integration of agents and automation systems," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*, pp. 2927–2934, IEEE, 2018.

[127] J. Smids, S. Nyholm, and H. Berkers, "Robots in the workplace: a threat to—or opportunity for—meaningful work?," *Philosophy & Technology*, vol. 33, no. 3, pp. 503–522, 2020.

[128] E. Wadsworth and D. Walters, "Safety and health at the heart of the future of work: Building on 100 years of experience," 2019.

[129] C. Giffi, P. Wellener, B. Dollar, H. A. Manolian, L. Monck, and C. Moutray, "Deloitte and the manufacturing institute skills gap and future of work study," *Deloitte Insights*, 2018.

[130] R. Zeng, Y. Wen, W. Zhao, and Y.-J. Liu, "View planning in robot active vision: A survey of systems, algorithms, and applications," *Computational Visual Media*, pp. 1–21, 2020.

[131] Cognilytica, "Data engineering, preparation, and labeling for ai," 2019.

[132] Y. Roh, G. Heo, and S. Whang, "A survey on data collection for machine learning: A big data - ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 10 2019.

[133] C. Éric Noël Laflamme, F. Pomerleau, and P. Giguère, "Driving datasets literature review," 2019.

[134] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo, "Openml: networked science in machine learning," *ACM SIGKDD Explorations Newsletter*, vol. 15, no. 2, pp. 49–60, 2014.

[135] H. Choi, C. Crump, C. Duriez, A. Elmquist, G. Hager, D. Han, F. Hearl, J. Hodgins, A. Jain, F. Leve, C. Li, F. Meier, D. Negrut, L. Righetti, A. Rodriguez, J. Tan, and J. Trinkle, "On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward," *Proceedings of the National Academy of Sciences*, vol. 118, no. 1, 2021.

[136] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1310–1319, 2017.

[137] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *CoRL*, 2018.

[138] K. Wang, F. Shi, W. Wang, Y. Nan, and S. Lian, "Synthetic data generation and adaption for object detection in smart vending machines," *CoRR*, vol. abs/1904.12294, 2019.

[139] M. Bertolini, D. Mezzogori, M. Neroni, and F. Zammori, "Machine learning for industrial applications: A comprehensive literature review," *Expert Systems with Applications*, vol. 175, p. 114820, 2021.

[140] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, 03 2016.

[141] J. Anderson, "Methods and applications of synthetic data generation," 2021.

[142] M. Stenmark and P. Nugues, "Natural language programming of industrial robots," in *IEEE ISR 2013*, pp. 1–5, 2013.

[143] S. Sanchez, H. Romero, and A. Morales, "A review: Comparison of performance metrics of pretrained models for object detection using the tensorflow framework," in *IOP Conference Series: Materials Science and Engineering*, vol. 844, p. 012024, IOP Publishing, 2020.

[144] "Ros-industrial." https://rosindustrial.org/about/description/.

[145] "Understanding the importance of 3d hand-eye calibration," tech. rep.

[146] "Concepts moveit!." http://moveit.ros.org/documentation/concepts/.

[147] I. A. Sucan, M. Moll, and E. Kavraki, "The open motion planning library," vol. 19, no. 4, pp. 72–82, 2012.

[148] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *IEEE International Conference on Robotics and Automation*, 2011.

[149] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," vol. 12, no. 4, pp. 566–580, 1996.

[150] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125, IEEE, 2018.

[151] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[152] 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. 5998–6008, 2017.

[153] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," *Robotics: Science and Systems VIII*, pp. 1–8, 2012.

[154] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, *et al.*, "Sim2real in robotics and automation: Applications and challenges," *IEEE transactions on automation science and engineering*, vol. 18, no. 2, pp. 398–400, 2021.

[155] S. Yang, X. Yu, and Y. Zhou, "Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example," in *2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, pp. 98–101, IEEE, 2020.

[156] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 12, p. 124003, 2021.

[157] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, *et al.*, "A generalist agent," *arXiv preprint arXiv:2205.06175*, 2022.

[158] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[159] "Schmalz." `https://www.schmalz.com/en/vacuum-technology-for-automation/vacuum-components/vacuum-generators/vacuum-generators-end-of-arm/handling-sets-ecbpi-308297/10.03.01.00504/`. Accessed: 2023-04-17.

[160] "Ur5 collaborative robot arm." `https://www.universal-robots.com/products/ur5-robot/`. Accessed: 2023-04-17.

[161] M. Quigley, "Ros: an open-source robot operating system," in *IEEE International Conference on Robotics and Automation*, 2009.

[162] "Smacc – state machine asynchronous c++." `https://smacc.dev/`. Accessed: 2023-04-17.

[163] "Github - ros/executive_smach: A procedural python-based task execution framework with ros integration.." `https://github.com/ros/executive_smach`. Accessed: 2023-05-02.

[164] M. Colledanchise and P. Ögren, "How Behavior Trees Modularize Hybrid Control Systems and Generalize Sequential Behavior Compositions, the Subsumption Architecture, and Decision Trees," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 372–389, 2017.

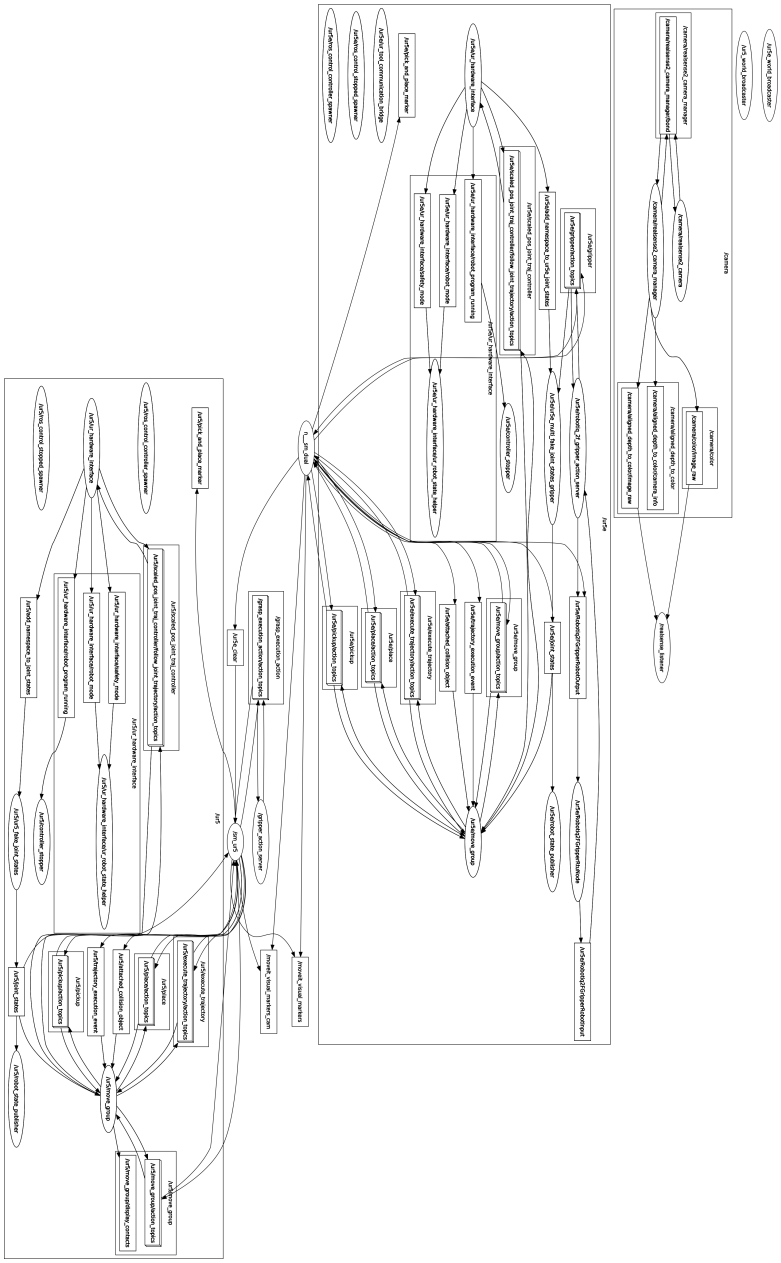
[165] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[166] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *ArXiv*, vol. abs/1404.3785, 2014.

APPENDICES

Transformation tree of the demonstration: Multi-robot collaboration for bin-picking task

Computational graph of the demonstration: Multi-robot collaboration for bin-picking task



Jānis Ārents was born in 1994 in Ērgļi. He received a professional Bachelor's degree in Electrical Engineering in 2017 and a professional Master's degree in Electrical Engineering in 2018 from Riga Technical University. Since 2016, he has been an electronics engineer, later a scientific assistant at the Institute of Electronics and Computer Science, where he is currently a researcher, contributing to various national and European projects. He was involved in the development of the "Smart robot with advanced vision, sensing, and human gesture understanding capabilities", which was evaluated by the Latvian Academy of Sciences as one of the most significant scientific achievements in Latvia in 2022. His scientific interests include smart robotic systems and their usage in the automation of various processes.