Generic Learning from Demonstration Approaches for Programming Collaborative Robots

El Zaatari, Shirine

Award date:
2022

Awarding institution:
Coventry University

Generic Learning from Demonstration Approaches for Programming Collaborative Robots

By

Shirine El Zaatari

May 2021

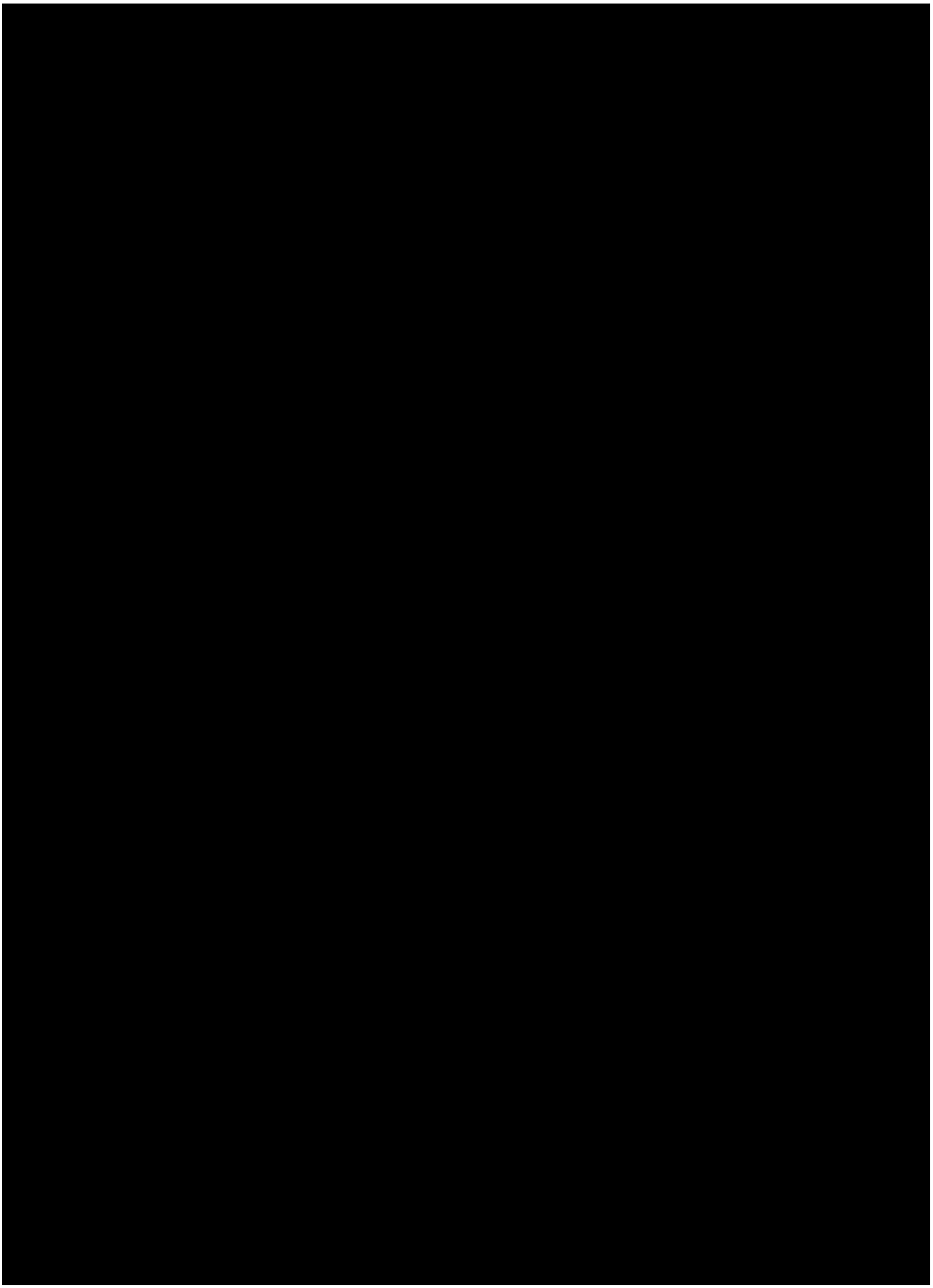
A thesis submitted in partial fulfilment of the University's requirements for the Degree of Doctor of Philosophy

Generic Learning from Demonstration Approaches for Programming Collaborative Robots

By

Shirine El Zaatari

May 2021



Ethical Approval

Coventry University

Certificate of Ethical Approval

Applicant:

Shirine El Zaatari

Project Title:

Developing an autonomous cobot system for shared industrial task

This is to certify that the above named applicant has completed the Coventry University Ethical Approval process and their project has been confirmed and approved as Medium Risk

Date of approval:

28 June 2018

Project Reference Number:

P63889

Table of Contents

List of Figures

List of Tables

Abbreviations

LfD	Learning from Demonstration
TP-LfD	Task Parametrized Learning from Demonstration
GMM	Gaussian Mixture Model
GMR	Gaussian Mixture Regression
OF	Oriented Frame
OLF	Orientation-less Frame
RL	Reinforcement Learning
SURF	Speeded-Up Robust Features
HRC	Human-Robot Collaboration
TP-GMM	Task Parametrized Gaussian Mixture Model
TP-GMR	Task Parametrized Gaussian Mixture Regression
IMU	Inertial Measurement Unit
DoF	Degree of Freedom
CoP	Centre of Pressure
CNN	Convolutional Neural Network
NMPC	Non-linear Motion Predictive Controller
FSM	Finite State Machine
PAD	Portable Assembly Demonstration
DTW	Dynamic Time Warping
GUI	Graphic User Interface
UAV	Unmanned Aerial Vehicle
EM	Expectation Maximisation
SVM	Support Vector Machine

Symbols

M	Total number of demonstrations; index m
T	Total number of data points; index t
D	Number of dimensions of data
P	Total number of task parameters; index p
K	Total number of Gaussian components; index k
$\xi_{m,t}$	Data point of demonstration m at time step t observed from the global frame
$b_{p,m}$	Position vector of frame p in demonstration m
$A_{p,m}$	Rotation matrix of frame p in demonstration m
$X_t^{p,m}$	Data point of demonstration m at time step t observed from frame p
X_t^p	Vector of data points at time step t observed from frame p for all demonstrations
$\mu_{p,k}$	Mean of Gaussian component k for frame p
$\Sigma_{p,k}$	Covariance matrix of Gaussian component k for frame p
π_k	Mixing coefficient of Gaussian component k
$\mathcal{N}$	Normal distribution
$\gamma_{t,k}$	Log likelihood of data at time step t to be in component k
$h_k(\xi_t^I)$	Weight of input part of ξ to be in component k
ξ_t^O	Output part of ξ
ξ_t^I	Input part of ξ
$\Sigma_{p,t}$	The covariance of the Gaussian (after regression) at time step t for frame p
g	Gripper state (open or closed)
x,y,z	3D real-life position of the end effector
$\{x,y\}_{p,m}$	Position of the frame p in demonstration m (pixel or real-life position)
$\{x_{cam\_pos}, y_{cam\_pos}, z_{cam\_pos}\}$	2D real-life position of the camera
$\{resol_x, resol_y\}$	Resolution of the image
α_{proj}	Perspective angle of the camera lens
$\{real_x, real_y\}$	Real length & width of table/surface displayed in the image
$d_{jp,m}$	Relative distance between frame j and p in demonstration m
$a_{jp,m}$	Relative angle between frame j and p in demonstration m
$\mu(d_{jp})$	Mean of relative distances across all demonstrations
$\sigma(d_{jp})$	Mean of relative angles across all demonstrations
ε	Threshold for redundancy condition

Redund	Matrix mapping redundancy relationship between frames
nbRelev	Number of relevant frames
probRelev	Probability of a frame being relevant
r	Reward
iterTot	Total number of iterations in RL algorithm; index *iter*
cost	Cost measuring reproduced path's similarity to demonstration path
iterPeriod	Number of iterations in a period
π	Policy
$\Delta probRelev_{iter}$	Disturbance in relevancy probability in an iteration
$\lambda probRelev$	The resultant relevancy probability to be used in an iteration after the disturbance
L_{tk}	Mean of the likelihoods that points at time step t belong to Gaussian k across all demonstrations
S	(Size of the kernel -1) / 2
ω	Scale of adjusting the likelihoods by their average
$[v_1,v_2]$	Eigen vectors of a Gaussian
$[e_1,e_2]$	Eigen values of a Gaussian
C_{ab}	An element in the covariance matrix describing the change of a with respect to b
$r_{t,p}^m$	The distance between point of time step t in demonstration m and a frame p
$\mathcal{N}_{(r)}(\mu_{(r)p,k}, \Sigma_{(r)p,k})$	A Gaussian component k modelling a portion of the $r_{t,p}^m$'s in a GMM.
$point_{pp',t}$	The point on the ring of frame p that is closest to frame p', at time step t

List of Publications

Journal Articles

1. El Zaatari, S., Marei, M., Li, W., & Usman, Z. (2019). 'Cobot programming for collaborative industrial tasks: An overview', *Robotics and Autonomous Systems*, *116*, pp: 162–180. doi: 10.1016/j.robot.2019.03.003.
2. El Zaatari, S., Wang, Y., Hu, Y., & Li, W. (2021). 'An Improved Approach of Task-Parameterized Learning from Demonstrations for Cobots in Dynamic Manufacturing', *Journal of Intelligent Manufacturing*. doi: 10.1007/s10845-021-01743-w.
3. El Zaatari, S., Wang, Y., Li, W., Peng, Y. (2021) 'iTP-LfD: Improved task parametrised learning from demonstration for adaptive path generation of cobot', *Robotics and Computer-Integrated Manufacturing*, 69, 102109. doi: 10.1016/j.rcim.2020.102109.
4. El Zaatari, S., Li, W., & Usman, Z. (2021). 'Ring Gaussian Mixture Modelling and Regression for Collaborative Robots', *Robotics and Autonomous Systems*, 145, 103864. https://doi.org/10.1016/j.robot.2021.103864

Conference Proceedings

1. El Zaatari, S. and Li, W. (2019) 'Visual Features as Frames of Reference in Task-Parametrised Learning from Demonstration', in *UK-RAS19 Conference*. Loughborough, UK, pp. 94–97. doi: 10.31256/UKRAS19.25.
2. El Zaatari, S., and Li, W. (2019). 'Reinforcement Learning to Identify Optimal Frames of Reference in Task-Parametrised Learning from Demonstration', in *20th Towards Autonomous Robotic Systems (TAROS).* London, UK.
3. El Zaatari, S., Wang, Y., Li, W. (2021) 'Reinforcement Learning based Optimization for Cobot's Path Generation in Collaborative Tasks', in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design.* Dalian, China. (Won Best Student Paper Award)

Co-authored Articles

1. Wang, Y., Hu, Y., El Zaatari, S., Li, W., Zhou, Y. (2021) 'Optimised Learning from Demonstrations for Collaborative Robots', *Robotics and Computer-Integrated Manufacturing*, 71, 102169. doi: 10.1016/j.rcim.2021.102169.
2. Marei, M., El Zaatari, S. and Li, W. (2021) 'Transfer learning enabled convolutional neural networks for estimating health state of cutting tools', *Robotics and Computer-Integrated Manufacturing*, 71, 102145. doi: 10.1016/j.rcim.2021.102145.

Abstract

The Industry 4.0 revolution is transforming the manufacturing scene to cater for new mass customisation demands and competitive production requirements. One of the ways Industry 4.0 aims to improve manufacturing is by adopting human-robot collaboration (HRC) due to its benefits in improving ergonomics, speeding production and merging human dexterity with robotic precision. Collaborative robots (cobots) are a stepping stone towards enabling HRC due to their intuitive programming interfaces, intrinsic safety features and agility. Cobots are marketed as "easily deployable" and "intuitively programmed". However, this is only true for a limited range of HRC applications in which the tasks of the human and the cobot are independent and the cobot follows a fixed path.

In order to expand the applicability of cobots to more complex HRC tasks in which the cobot has to be human-aware and exhibit flexible behaviour, we explored a new programming technique, namely task-parametrized learning from demonstration (TP-LfD). In TP-LfD, the cobot is shown a few demonstrations of a task conducted under changing circumstances and a regressed model of the task is learnt to support new previously-unseen circumstances. Using TP-LfD requires the human teacher to specify the task-relevant objects and the methods of detecting and localising the objects. This is a time consuming process subject to human errors. To address the above challenge, in this thesis, we present a generic solution that automatically detects and optimises the choice of task parameters for TP-LfD. We evaluated our solution in multiple simulation industrial tasks where positions of objects vary in the scene.

In this thesis, Gaussian mixture models (GMM) are used to model tasks using TP-LfD. A GMM is generated to model the path with respect to each task parameter. Each task parameter is a frame of reference characterised by a position and orientation. GMMs intrinsically vary when the orientation of a frame changes. However, in some cases, such as pick-and-place, the orientation of a frame of reference is irrelevant to the task. Therefore, in this thesis, we designed a new model, called the ring Gaussian, which models paths with respect to frames whose orientation is task-irrelevant. Our model generated more efficient and successful paths compared to the traditional GMM model. The proposed previous two contributions were integrated in one end-to-end algorithm to program cobots for a wide range of HRC industrial tasks, including pick-and-place and handover. The performance of the robot was more efficient and accurate than when either of the contributions was not used. In addition, case studies in this thesis show how the integrated algorithm can help overcome common problems such as partial occlusion and obstacle avoidance. The outcomes of this thesis were presented in 4 peer-reviewed journal papers and an open source code.

To conclude, this thesis contributes towards the intuitive programming of cobots for flexible manufacturing tasks. Simulated industrial tasks and case studies in this thesis demonstrated the

successful use of the developed algorithm to program cobots for tasks with changing object position and human involvement, without the need for programming expertise.

Keywords: Human-robot collaboration, Industrial robots, Learning from demonstration, Task-parametrized Gaussian mixture models

Declaration

I hereby declare that no portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institutes of learning.

Acknowledgments

I am eternally grateful, first and foremost to God for providing the strength and support to undertake this educational journey. I pray God accepts it in my good deeds, and blesses it so it benefits His creations.

I would like to thank my Director of Studies, Professor Weidong Li for his kind and valuable support throughout my PhD journey. Professor Li aided me in overcoming challenging phases in my PhD and provided valuable life and career advice. I couldn't have wished for a better supervisor. Moreover, I am grateful to Yuqi Wang who was of great help during my experimentation process.

I would also like to thank my ex-supervisor Dr. Zahid Usman, who made sure I started this PhD on the right foot, as well as my supervisors Dr. Nazaref Shah and Professor Kuo Ming Chao for their helpful comments and advice. Moreover, Yudie Hu and Yiqun Peng also provided great aid in my experiments.

I owe my success to my parents, Mama and Baba, who have made the greatest sacrifice for my education and wellbeing: their time, wealth, health, essentially, their life. I dedicate my work to you both and I pray God rewards you, because nothing I do will be enough.

I couldn't have done my PhD without my husband, Anees Anwar, who was my emotional counsellor, technical consultant and health coach all at once. Our family, my mother-in-law, father-in-law, sisters-in-law for always supporting me and pushing me forward. Special thanks goes to my little boy, Ibrahim, who came as a little blessing during my third year of PhD. He increased my sense of urgency to finish my studies.

Last but not least, I couldn't have done this without my friends, Dr. Sahar Shahid and Dr. Lara Carballo who were like big sisters to me, always listening to me and giving me the best guidance in my personal and academic life. My friends from home, Sara, Rewa, Abir, Nisreen, Lama and Rawan who continuously provided me with motivation and strength from far away. I thank my lab friends, Dr. Romali Biswal, Zeina Naboulsi, Dr. Ibrahim AlMakky, Anees Abu-Monshar, Dr. Shamir Tahir, Dr. Rohit Kshirsagar, Dr. Mohamed Marei and Dr. Sattam Saha for the good times we had in between our studies, for the laughs and tea breaks. Thank you for the lovely company.

Dedication

I dedicate this work to my mother and father, who have invested everything they have in me and my education. They always encouraged me and supported my decisions. I hope I become as good of a parent as you are.

Chapter 1: Introduction

1.1. Motivation

1.1.1. Industry 4.0

Originating from Germany in 2011, Industry 4.0 is a global effort to revolutionise the manufacturing sector. Industry 4.0 aims to integrate the latest advancements in technology such as artificial intelligence, cloud computing, simulation and augmented/virtual reality to improve factories. These improvements aim to enable mass customisation, facilitate communications between machines, optimise operations and enable human-robot interactions (Masood & Sonntag, 2020). Research groups have been studying the aforementioned latest technological advancements and the best ways to use them to improve industrial settings.

1.1.2. Benefits of HRC

Human-robot collaboration (HRC) is one of the main ventures being researched in the efforts of pushing Industry 4.0 forward. HRC refers to application scenarios where a robot, usually a collaborative robot (cobot), and a human occupy the same workspace and interact to accomplish collaborative tasks. This interaction is through turn-taking and part/tool passing as well as separately completing parts of the tasks best suited to the human/cobot's abilities (Haddadin & Croft, 2016). Human operators can deal with uncertainties and variability. Moreover, they can perform tasks requiring high dexterity. On the other hand, cobots offer advantages in terms of repeatability, speed, stamina and physical strength. In HRC, the benefits of automation and manual labour are combined (Müller et al., 2016), which may lead to an overall improvement in production time, efficiency, ergonomics and a reduction in cost and factory floor space.

1.1.3. Market Cobots

Cobots are available in the market such as Universal Robotics (UR), ABB's YuMi, KUKA's LBR iiwa, etc. (ROBOTIQ, 2020). They are equipped with a range of special features such as force torque sensors, no trap points and intuitive programming user interfaces (UIs) that make them easily deployable to work alongside human operators. Moreover, most are safety certified according to safety standards ISO 10218 for industrial robots and/or ISO/TS 15066/2016 for collaborative robots (ROBOTIQ, 2020).

1.1.4. Industrial Implementations

HRC is being increasingly and successfully implemented in several industrial setting. For example, in the BMW Mini factory, an operator places rivets on a part and a cobot fastens the rivets while the operator continues to perform other tasks simultaneously on the part (Reeco, n.d.). Nissan's large-scale Yokohama plant deployed UR10 cobots to loosen bolts and carry heavy components to relieve the workforce of these arduous tasks and speed up the manufacturing process (Carrus Home, n.d.). Skoda

also introduced a KUKA cobot to work alongside humans in the production of direct-shift-gearboxes (KUKA, n.d.). Therefore, the industrial interest in cobots and their proved benefits encouraged us to research solutions for easier and improved cobot deployment.

1.2. Human-Robot Collaboration

1.2.1. Levels of Collaboration

HRC tasks can be divided into levels of collaboration according to the relation between a cobot, an operator, work piece(s) and the process(s) being performed on the work piece(s). That is, the levels are defined according to the degree of task intersection and dependency between the operator and the cobot (Cesta et al., 2016). The levels are the following (Figure 1.1):

- Independent: An operator and a cobot operate on separate work pieces (W_1 and W_2 illustrated in Figure 1.1) independently for their individual manufacturing processes (P_1 for W_1 and P_2 for W_2). The collaborative element is due to the co-presence of the operator and cobot in the same workspace without a fence or guard. That is, safety is achieved through the cobot's intrinsic safety and/or added hardware/software safety elements, such as collision avoidance. Therefore, the cobot is aware of the operator's presence and acts safely.
- Simultaneous: An operator and a cobot operate on separate processes (P_1 and P_2 respectively) on the same work piece (W) at the same time. There is no time or task dependency between them. However, the cobot needs to be spatially aware of the operator and his/her task requirements in order to respect the operator's space. Being able to concurrently operate on the work piece will minimise the transmit time of the work piece between the cobot and human, thereby improving productivity and space utilisation.
- Sequential: An operator and a cobot perform sequential manufacturing processes (P_1 and P_2) on the same work piece. There are time dependencies between the cobot and operator for their processes. For instance, the cobot works on P_1 for the work piece as an input to support the operator to carry on P_2 for the work piece. In most cases, the cobot is arranged to handle tedious processes to improve the operator's working conditions.
- Supportive: An operator and a cobot work towards the same process (P) on the same work piece (W) interactively. There is dependency between the actions of the cobot and the operator. That is, without one, another cannot perform the task. The cobot needs to understand the operator's intent and the task requirements in order to provide appropriate assistance. The role of the cobot is to physically assist the operator with work pieces which improves ergonomics.

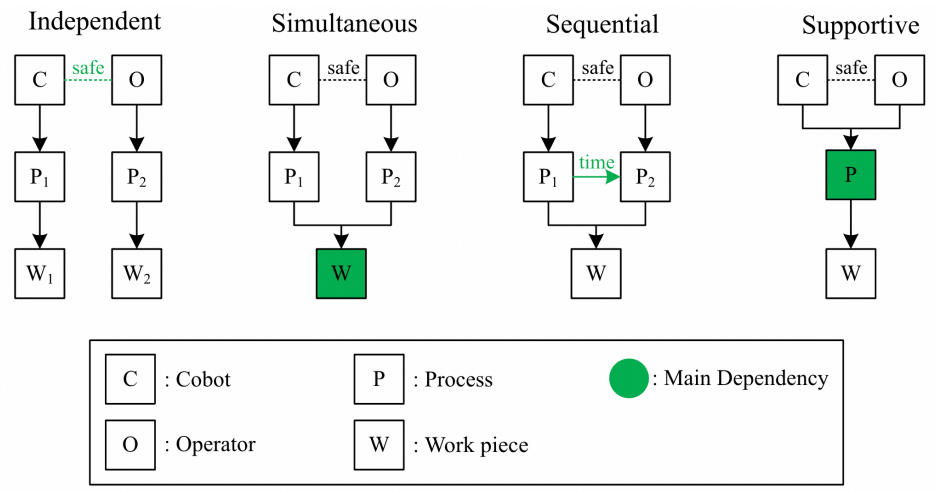


Figure 1.1 Levels of human-robot collaboration

1.2.2. Limitations of Market Cobots

Upon trying to program market cobots for the different HRC levels, one is faced with many challenges. To begin with, even though market cobots are safety certified, risk assessment is still required prior deployment. That is because even with the intrinsic speed limitations, if the cobot comes in contact with a sensitive part of the human body, e.g. the head, it is dangerous. Therefore, additional safety features are required such as collision avoidance. Moreover, the built-in UIs is capable of intuitively programming the cobot by specifying way points. However, it does not cater for flexible behaviour which is a main characteristic of HRC tasks. Therefore, it is found that the current industrial implementations of HRC, such as the ones mentioned in Section 1.1.4, are limited in complexity. The cobot, even though is working alongside a human, only adheres to a predetermined path with very limited flexibility.

To fully reach the potential of HRC, cobots need to be provided with two main capabilities: intuitive programming and flexible behaviour. Intuitive programming is important so that operators on the shop floor can quickly adjust cobot programs to suit preference, task changing, and understand how the cobot "thinks" to be able to interact with it safely and efficiently. Flexible behaviour is important so that the cobot can act in unpredictable situations in which the human is adding uncertainty. That is why, researchers are focused on integrating intelligent communication, optimisation and learning algorithms with cobots to expand their application range. Chapter 2 elaborates on those technologies.

1.3. Task Parametrized Learning from Demonstration

1.3.1. Overview

In an effort to provide cobots with the ability to be intuitively programmed to provide flexible behaviour, task parametrized learning from demonstration (TP-LfD) was found to be an effective algorithm. TP-LfD takes a set of demonstrations of a cobot acting in a few varied settings, and then it generalises over them and reproduces a new path for a new setting (Calinon, 2016). The main steps of

TP-LfD are the following, with an illustrative example, in which a piece of debris needs to be brushed onto a dustpan:

1. Recording demonstrations: Figure 1.2 shows an illustrative example of four user-provided demonstrations for brushing a debris onto a dustpan. Both objects have variable positions. Each demonstration consists of an image of the initial setup and a demonstration path in which the debris is brushed by a cobot onto the dustpan.
2. Detecting task parameters: In TP-LfD, the path of a cobot is encoded with respect to multiple frames of references (task parameters). Task parameters are usually associated with locations of task-relevant objects or locations. For example, in Figure 1.2, the debris and dustpan are each associated with a frame of reference, and each of them is defined by a location vector b and orientation α with respect to an arbitrary global coordinate system.

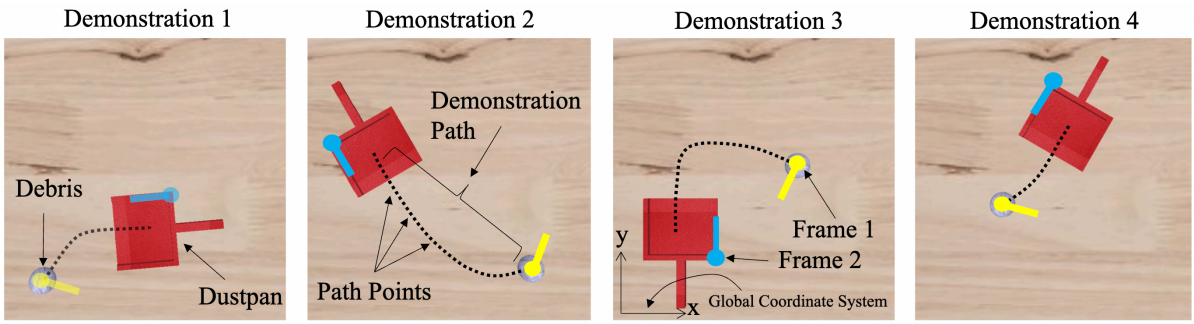


Figure 1.2 Four demonstration recording of the cobot's path for brushing the debris onto the dustpan. A frame of reference is associated with each object. Each frame is defined by location vector b and orientation α with respect to a global frame of reference.

3. Encoding the paths with respect to the task parameters: A path can be modelled as task parameterised Gaussian mixture model (TP-GMM) from the perspective of each available frame. In this thesis, GMMs are chosen to model the paths according to the central limit theorem, the distribution of the sample means of a population approaches a normal distribution as the sample size gets larger. That implies that a normal distributions can successfully model many complex systems with the least amount of prior knowledge (Goodfellow et al., 2016). In Figure 1.3 (left), Frames 1 from all the four demonstrations shown in Figure 1. are aligned as if the paths are observed from Frame 1. Each ellipse is a Gaussian probabilistic distribution (in the thesis, each ellipse is called a Gaussian for simplicity). The paths are encoded with three (as an example) Gaussians (1.1, 1.2 and 1.3) for each one third of the path. In Figure 1.3 (right), a similar process is done for Frame 2 resulting in Gaussians 2.1, 2.2 and 2.3.

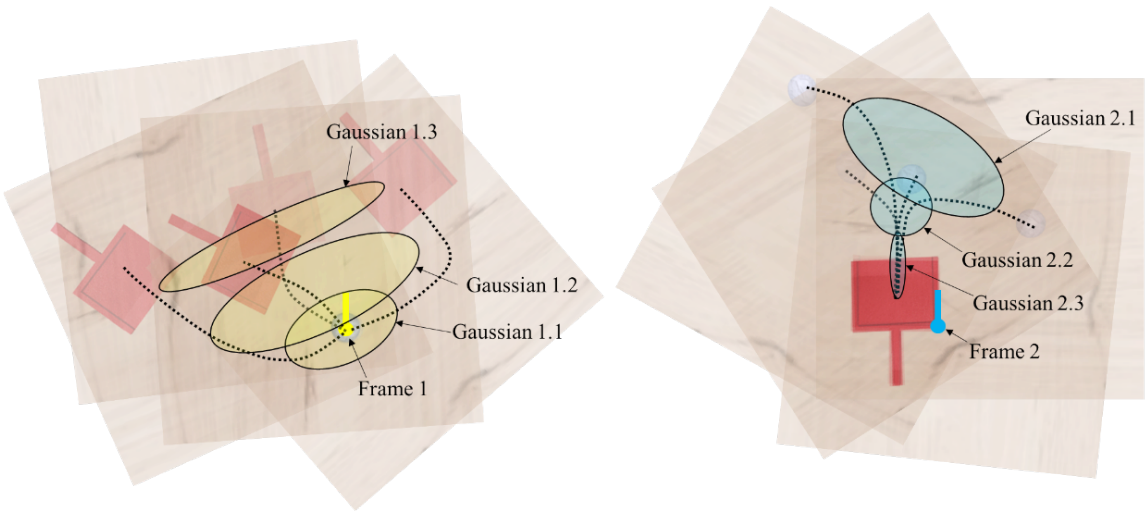


Figure 1.3 GMMs, constituting of 3 Gaussians each, encoding the path as observed from Frame 1 and Frame 2.

4. Reproducing new path: Based on the demonstrations, as the positions and orientations of the dustpan and the debris change, the cobot should reproduce a new path for this new scenario accordingly (Figure 1.4(a)). Obtained GMMs for each frame are aligned depending on the frames' new positions and orientations in this scenario (Figure 1.4(b)). For each frame, GMR is performed, where a path point distribution, i.e. Gaussian, for each time step t is sampled from the GMM (Gaussians under different time steps are shown in Figure 1.4(c)). Then, for every time step t, a product of Gaussians is performed between the Gaussians of both frames. The product results in a new Gaussian, whose mean is the reproduced path point at time t (Figure 1.4(d)). The weight for each Gaussian in the product is equivalent to the inverse of its covariance matrix. Thus, distributions that have higher covariance possess less priority in the sum. In reality, this means that when demonstrations have a consistent path (small covariance) with respect to a certain object (frame of reference), the cobot will learn that the consistency of this path is significant and needs to be maintained in future reproductions. However, when the demonstration does not have a consistent path with respect to a certain object, the cobot is more flexible with respect to this object in future reproductions. This reproduction process is known as task parametrized Gaussian mixture regression (TP-GMR).

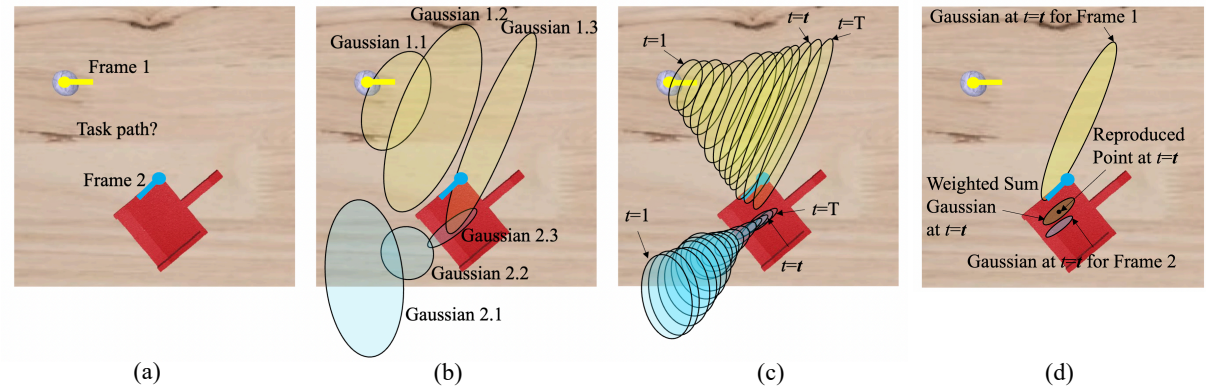


Figure 1.4 The main steps of the task parametrized Gaussian mixture regression algorithm. (a) Detect frames in their new positions. (b) Align their GMMs in the new position. (c) Regress the GMM to obtain a Gaussian for every time step t. (d) Add Gaussians of all frames from time step t, to obtain reproduced path point.

1.3.2. Limitations of TP-GMM/R

TP-GMM and TP-GMR (TP-GMM/R) are successfully used to program cobots in multiple research works, such as in (Alizadeh & Saduanov, 2017; Y. Huang et al., 2018; Rozo et al., 2016). One of the main prerequisites for the success of the TP-GMM/R algorithm is the accurate detection of the task parameters. In most cases, task parameters take the form of frames of reference describing the position and orientation of task-relevant objects in space. To program each task using TP-GMM/R, the programmer should ensure that the algorithm is detecting task-relevant objects. This requires tailored computer vision algorithms done by a professional programmer, which is time and resource consuming. This means that TP-GMM/R is no longer an intuitive algorithm that can be applied to program tasks without programming expertise. To combat this setback, in this thesis, a generic vision algorithm was created that automatically detects task-relevant frames of reference.

Moreover, due to the nature and shape of the GMM, the path points are modelled with respect to the task parameters such that when a task parameter's orientation changes, the GMM position changes. That means, the path points are affected by the orientation of task parameters. However, this might not accurately reflect task requirements, since in most cases, the orientation of a task parameter, does not affect the general direction of the task path but only the gripping orientation. Therefore, in this thesis, a new model was created, called the ring Gaussian, which accurately encodes path points without accounting for the orientation of task parameters.

1.4. Research Aims and Objectives

With the fast advancement and increased competitiveness of industrial parties, many are seeking to increase production efficiency and reduce costs by employing collaborative robots. Collaborative robots are meant to be easily deployable and movable around the factory floor. This requires them to be safe to run alongside human operators without requiring expensive fence setups. The aim of this thesis is to make collaborative robots easily programmable with minimal programming experience to

facilitate their deployment and movement in a factory floor. Moreover, the collaborative robot needs to be equipped with the ability to function in unpredictable environments. These aims are accomplished by setting off to accomplish the following objectives:

- Detect and identify task parameters for TP-GMM without requiring custom computer vision algorithms.
- Improve the performance of TP-GMM algorithm in common industrial tasks.
- Encourage the usage of TP-GMM to program collaborative robots for industrial tasks.

1.5. Thesis Contributions

The contributions in this thesis can be divided into three categories, in the field of using learning from demonstration to program cobots for industrial tasks:

1. Create the idea and develop the pipeline of using generic visual features as task parameters for TP-GMM. A MATLAB code was written to automatically detect and optimise task parameters for TP-GMM/R, assuming task parameters are positions and orientations of task-relevant objects in space, which is generally the case. To achieve this, we extract generic visual features such as Speeded Up Robust Features (SURF) from demonstration images. Since a large number of SURF features will be obtained, not all of which are task-relevant, the performance of the TP-GMM/R will deteriorate if all features are used as task parameters. Therefore, 1) a statistical algorithm groups redundant visual features, i.e. features belonging to the same object, together; 2) a reinforcement learning algorithm is used to identify the optimal features to be used as task parameters in TP-GMM/R, i.e. to eliminate features that are irrelevant to the task. The algorithm was tested in CoppeliaSim simulation environment for 4 different industrial tasks. This is discussed in chapter 3.
2. Define orientation-less frames and identify the problem that TP-GMMs do not cater well for them. Create a new model, the ring Gaussian, that accurately caters for frames whose orientation is irrelevant for task paths, i.e. orientation-less frames. To achieve this, a MATLAB code was written to 1) automatically identify orientation-less frames based on criteria that we defined; 2) calculate the ring Gaussians for each orientation-less frame; 3) make adjustments on the TP-GMR algorithm to be able to cater for the new ring Gaussian model. This is discussed in chapter 4.
3. Integrate the above two works in an end-to-end tool to intuitively program cobots for industrial tasks. 1) A tutorial document is created to teach users with no programming experience how to use the software. This tool is used to overcome two industrial task challenges: partial occlusion and obstacle avoidance. 2) Partial occlusion is overcome by utilising the redundant frames belonging to the same object. 3) Obstacle avoidance is

performed by utilising the ring Gaussians calculated for orientation-less frames. This is discussed in chapter 5.

1.6. Research Outputs

As a result of the research conducted for this PhD thesis, the following articles have been produced:

1. El Zaatari, S., Marei, M., Li, W., & Usman, Z. (2019). 'Cobot programming for collaborative industrial tasks: An overview', *Robotics and Autonomous Systems*, *116*, 162–180. doi: 10.1016/j.robot.2019.03.003.
2. El Zaatari, S. and Li, W. (2019) 'Visual Features as Frames of Reference in Task-Parametrised Learning from Demonstration', in UK-RAS19 Conference. Loughborough, UK, pp. 94–97. doi: 10.31256/UKRAS19.25.
3. El Zaatari, S., and Li, W. (2019). 'Reinforcement Learning to Identify Optimal Frames of Reference in Task-Parametrised Learning from Demonstration', in 20th Towards Autonomous Robotic Systems (TAROS). London, UK.
4. El Zaatari, S., Wang, Y., Hu, Y., & Li, W. (2021). 'An Improved Approach of Task-Parameterized Learning from Demonstrations for Cobots in Dynamic Manufacturing', *Journal of Intelligent Manufacturing*. doi: 10.1007/s10845-021-01743-w.
5. El Zaatari, S., Wang, Y., Li, W., Peng, Y. (2021) 'iTP-LfD: Improved task parametrised learning from demonstration for adaptive path generation of cobot', Robotics and Computer-Integrated Manufacturing, 69, 102109. doi: 10.1016/j.rcim.2020.102109.
6. El Zaatari, S., Li, W., & Usman, Z. (2021). 'Ring Gaussian Mixture Modelling and Regression for Collaborative Robots', *Robotics and Autonomous Systems*. (Revision)
7. El Zaatari, S., Wang, Y., Li, W. (2021) 'Reinforcement Learning based Optimization for Cobot's Path Generation in Collaborative Tasks', in *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design.* Dalian, China. (Won Best Student Paper Award)

Our contribution in previously mentioned papers was to research, create, code and troubleshoot the algorithms, perform tests on simulation robots and real-life images as well as write and review the papers. Moreover, the following articles have been co-authored:

1. Wang, Y., Hu, Y., El Zaatari, S., Li, W., Zhou, Y. (2021) 'Optimised Learning from Demonstrations for Collaborative Robots', Robotics and Computer-Integrated Manufacturing, 71, 102169. doi: 10.1016/j.rcim.2021.102169.
2. Marei, M., El Zaatari, S. and Li, W. (2021) 'Transfer learning enabled convolutional neural networks for estimating health state of cutting tools', Robotics and Computer-Integrated Manufacturing, 71, 102145. doi: 10.1016/j.rcim.2021.102145.

1.7. Thesis Overview

This thesis is split into six chapters, including the introduction in this chapter. Chapter 2 provides a review of intelligent cobot programming technologies, split into three categories: communication, optimisation and learning (El Zaatari et al., 2019). Chapter 3 focuses on the first contribution in the thesis: an automated task parameter detector and optimiser for task parametrized learning from demonstration (El Zaatari, Wang, Hu, et al., 2021; El Zaatari,Wang, Li et al., 2021). Chapter 4 elaborates on the second contribution in the thesis: an innovative Gaussian model to improve the performance of task parametrized learning from demonstration when a frame of reference is orientation-less (El Zaatari, Li et al., 2021). Chapter 5 presents the integration of the contributions of chapter 3 and 4 together as well as a solution for partial occlusion and obstacle avoidance. Chapter 6 concludes the thesis and points towards future works to improve the presented methods and move cobots closer to practical applications.

Chapter 2: Background: Cobot Programming

2.1. Introduction

The programming process entails providing a cobot with the ability to understand the state of the environment and perform actions that advance the system towards a planned collaborative goal. Traditionally, a human, the programmer, is only involved off-line for an industrial robot program. These programs are inflexible and not human-aware, and cannot be altered during runtime, unless an error occurs and debugging is needed. Based on that, a robot functions in a deterministic environment in which an operator is not part of. However, in HRC, an operator adds stochasticity and unpredictability to the environment. The human involvement in the cobot's program goes beyond the programmer's traditional off-line role. The operator also becomes involved in the cobot's program during run-time, or on-line.

An operator can be involved in modifying or affecting a cobot's program either explicitly or implicitly. Explicit involvement occurs in the form of direct communication, i.e., the human sends information or instructions to the cobot. Implicit involvement occurs such that the cobot observes the human's states and alters its policy accordingly. The policy can be learnt from prior data or modelled manually by programmers. Based on these different modes of operator involvement, this chapter identifies three different programming technologies that give the cobot the ability to act flexibly and/or be programmed intuitively. These programming technologies are especially essential for Sequential and Supportive HRC scenarios. The programming technologies identified are:

- Communication: An operator controls a cobot through a communication channel that can be verbal (speech) or non-verbal. Non-verbal communication includes gestures, gaze, head pose, haptics and UIs. The off-line role of the programmer is to program and define possible cobot actions and the underlying motion control. The on-line involvement of the operator is mostly explicit, triggering the cobot into pre-defined discrete or continuous actions.
- Optimisation: Important aspects of a cobot's surroundings, such as obstacles and tool positions, are mathematically modelled as a function of the cobot's actions. Those form cost functions that are optimised to generate desirable performance. The cobot's program can be made to minimise an operator's workload, energy consumed and time wasted, or maximise physical comfort and trust, product quality, etc. During off-line development, the programmer designs cost functions and optimisation algorithms. During runtime, the operator usually impacts the cobot's performance implicitly, since he/she will be a part of a cost function. The advantage of this method is the higher likelihood of performing more optimally than a human operator.
- Learning: A cobot learns a skill similar to how a human would, e.g., through observing demonstrations, trial and error, receiving feedback and asking questions. The off-line role

of a programmer is to design the learning algorithm and provide initial data for the cobot to learn from. That could be in the form of demonstrations, trial-and-error iterations (resulting in a policy), training data, etc. During runtime, an operator might be able to explicitly affect the cobot's policy by providing additional data, such as feedback, answers to questions, personalised demonstrations, etc. Moreover, the operator might serve as a prior in the cobot's probabilistic learning algorithm, i.e., affecting the cobot implicitly by being part of the observed environment.

Different programming technologies enable different degrees and forms of cobot autonomy. As cobot autonomy increases, an operator is more likely to feel unease due to the cobot's decreased predictability. However, as the cobot autonomy decreases, the operator is required to make decisions on behalf of both, which increases the mental workload. Therefore, one programming technology is not strictly better than the other, but can be mixed to exploit their benefits while negating, or limiting, their drawbacks. The programming technology supported should also be chosen in light of industrial scenario complexity and the operator's knowledge of the task.

In this chapter, we elaborate on these programming technologies, their variations and implementation in HRC scenarios. Moreover, we delve deeper into TP-LfD, the main programming technique that this thesis utilises.

2.2. Communication

Humans rely heavily on communication to work in teams and complete tasks fluently and efficiently. Communication can be made to issue orders, convey intention and ask/answer questions. Researchers have been working on enabling communication between humans and cobots such that the human is able to command the cobot through different communication modes. Communication-based programming, where an operator commands or designs the cobot's program through communication mediums, gives a level of direct authority from the operator. Whether that is desirable or not depends on the complexity of the task, the knowledge of the operator and the industrial party's choice. However, it would certainly increase an operator's trust in a cobot and the willingness to work alongside it. This would aid in the introduction and normalisation of cobots in manufacturing. However, communication mediums vary drastically in intuitiveness and reliability. The works mentioned in this subsection are categorised by communication mode: body language and speech, user interfaces and haptics.

2.2.1. Body Language and Speech

Body language as a means of commanding a cobot includes using gestures, pointing, head pose, and gaze. Speech refers to uttering commands verbally. These two communication modes are combined in the same subsection since they share a similar algorithmic pipeline: First, a communication guideline must be defined, i.e. in what ways of language/words/gestures, will the operator communicate with the

cobot? Communication signals are detected, recognised and mapped to executable actions for a cobot. The rest of this subsection tackles the different research works done towards these different steps in the 'body language and speech' communication pipeline.

Communication Guidelines

Defining an effective communication guideline involves specifying usable communication signals, such as a set of gestures or phrases, and when and why to use them. Various approaches have been found in the literature to define a communication guideline. To begin with, a set of usable gestures or phrases can be predefined strictly in a fixed set. A gesture lexicon can be extracted from observing human-human interactions (Calisgan et al., 2012; Gleeson et al., 2013; Pohlt et al., 2017). However, gestures extracted from observing human-human interactions will not necessarily be easy to recognise and differentiate. That is because many of them tend to be very subtle, context-specific and sometimes person-specific.

In other cases, a set of gesturing rules is used to create a gesture set. Barattini et al. proposed a standard set of gestures for a given task (the gestures must be distinct from other task actions), and evaluated a gesture recognition algorithm on the proposed set (Barattini et al., 2012). However, even with optimally chosen gestures, having to memorise and adhere to a fixed set of signals can be mentally draining and unintuitive for the operator.

Allowing a human to communicate with a cobot in his/her own way results in more effective, natural and intuitive communication. Cheng et al. designed a framework to extract robotic operations from natural language based on relationships between mentioned work pieces and representing the relationships in matrix form (Cheng et al., 2018). She and Chai used interactive learning to learn verb semantics in a noisy incomplete environment (She & Chai, 2017). The cobot is capable of asking the right questions to learn required actions and corresponding objects, states and tools. Maurtua et al. also analysed natural language in light of task ontology to extract commands (Maurtua et al., 2017).

Generating a natural language system is challenging since the language use differs drastically as the operator progresses with work. Nakata et al. showed that in a collaborative task where only verbal communication is allowed, the frequency of morphemes (i.e. words belonging to these certain types: object, modifier, robot action, user action) decreases as the number of task trials increase (Nakata et al., 2011). That is because humans naturally start emitting words as they become accustomed to the task. They naturally start considering and accommodating their team-mates' needs without those needs being explicitly expressed. Kobayashi et al. also showed that the use of descriptive words decreases as the number of task trials increase (Kobayashi et al., 2020). Therefore, any language model between a human and a cobot should account for the change in human language as the human becomes more accustomed to the task.

Multi-modal Communication

Different research works have shown that communication modes can be concatenated in different ways for better context understanding. Using multi-modal communication can outperform single mode communication. Multi-modal communication can sometimes be complementary such as a point-and-command system. It can also be redundant such as a same-command speech and gesture system (Maurtua et al., 2017). The challenge lies in how to combine information from different communication modes to successfully draw conclusions. Srimal et al. used fuzzy logic to combine pointing gestures with speech in order to identify pointing targets or execute spatial commands (Srimal et al., 2017). Giuliani and Knoll used a score-based system to identify which action to perform on which object (Giuliani & Knoll, 2013). They represent an object, its corresponding action, and a score R as one tuple. When a speech or pointing command is uttered mentioning an object or an action, the scores of the tuples including the object or the action are increased. When the score exceeds a threshold, the action is executed on the object.

Maurtua et al. used a fusion engine to ensure voice and pointing commands are not contradictory and combine them into a single command output to the execution manager (Maurtua et al., 2017). They designed a gesture, including pointing, and voice command system with safety functions integrated and implemented it on a KUKA IIWA. Their system was rated promisingly in terms of naturalness, usefulness and reliability in an extensive study.

It is important to consider if/when multi-modal communication is needed, before considering how to implement it. Admoni et al. worked on determining when a pointing gesture is necessary along with a verbal description to identify an object on a table (Admoni et al., 2016). A gesture only be necessary, for example, if there were several objects of the same verbal description in close proximity. Their work can be used to guide and prompt communication to only when it is needed, which would improve efficiency and lessen chances of error and confusion.

From Communicated Signals to Executable Actions

After specifying a communication guideline, the permissible communicated signals must be mapped to executable cobot actions, i.e. a signal should be made a command. This can be done manually or through learning:

- Manually: A programmer manually assigns gestures to cobot actions off-line according to task needs (Barattini et al., 2012). Human-human interactions can help programmers understand which gestures map best to which actions (Gleeson et al., 2013; Pohlt et al., 2017). If the action domain is continuous such as in (Wongphati et al., 2015), then the gesture-action mapping is done through calibration.
- Learning: Interactive learning can be used so that an operator plays a role in the signal-action mapping. Shukla et al. taught a cobot required actions to perform given a gesture using

incremental human feedback (Shukla et al., 2017). However, this can present unnecessary complications in an industrial environment where insufficient variability in the mapping is expected. In a continuous action domain, such as in (Huang & Mutlu, 2016), the gaze-object associations are obtained by a pre-trained Support Vector Machine (SVM) in order to ultimately predict the human's intent (i.e. the object the human is looking at). This helps the cobot start acting towards the object before an explicit command is uttered. This is particularly useful when the communication channel domain is continuous, such as gaze direction, and requires segmentation before mapping.

Signal Recognition

Delving into the technicalities of signal recognition, whether it is gestures, speech, haptics, etc. is beyond the scope of this chapter and will only be discussed briefly, as numerous relevant reviews already exist. Readers are referred to (Liu & Wang, 2018) for an extensive review on gesture recognition technologies in light of industrial HRC. Similarly, Benzeghiba et al. provided a review on speech recognition technologies (Benzeghiba et al., 2007).

To recognise gestures, the human skeletal frame, or pose, must be detected. Modelling the gesture depends on whether it is static or dynamic. For a pose to be detected as a static, its temporal length needs to exceed a specific threshold (Pedersen et al., 2014). Then, the angles of the different segments of the human skeleton are thresholded to classify the gesture. In the case of dynamic gestures, Coupeté et al. modelled dynamic gestures as a Hidden Markov Model and classified them using K-means (Coupeté et al., 2019).

Table 2.1 shows the advantages and disadvantages of different sensing technologies for pose detection. Figure 2.1 shows examples of sensing technology for human gestures. Given recent advancements in deep learning pose detection algorithms, such as OpenPose (Zhe Cao et al., 2017) and Faster R-CNN (Nu

(a)

(b)

(c)

Modelling the gesture depends on whether it is static or dynamic. For a pose to be detected as a static, its temporal length needs to exceed a specific threshold (Pedersen et al., 2014). Then, the angles of the different segments of the human skeleton are thresholded to classify the gesture. In the case of dynamic gestures, Coupeté et al. modelled dynamic gestures as a Hidden Markov Model and classified them using K-means (Coupeté et al., 2019).

Table 2.1 Advantages and disadvantages of different gesture/pose detection sensors.

Sensor	**Advantages**	**Disadvantages**
3D-cameras, e.g. Kinect v2 (Kumičáková et al., 2017; Makrini et al., 2017) and ASUS Xtion PRO Live (Pedersen et al., 2014)	Non-intrusive, easy to setup	Restricted detection region, prone to occlusion, dependent on lighting conditions, detection algorithm dependent on 3D sensing output (point cloud, depth map...)
RGB cameras	Non-intrusive, easy and affordable to setup, availability of reliable algorithms	Restricted detection region, prone to occlusion, dependent on lighting conditions
IMU Jackets, e.g. (de Gea Fernández et al., 2017; Neto et al., 2019)	No occlusion, no dependency on lighting and environmental factors	Restrict mobility, not one-size-fits-all, does not measure hand gestures
Wrist bands, e.g. (Chen et al., 2007)	No occlusion, no dependency on lighting and environmental factors	Restrict mobility, difficult time-consuming setup
Motion Capture, e.g. (Vogt et al., 2017)	High accuracy, computationally effective, less prone to occlusion	Expensive and time-consuming setup, restricted detection region

Pointing gestures are different since they are related to the space and objects. Recognising or classifying them constitutes identifying the object or location being pointed at. In (Maurtua et al., 2017), Euclidean cluster extraction was used to detect the human's forearm. The forearm is modelled as a cylinder, and the pointing target is identified as the cylinder's axis intersection with the workspace. Srimal et al. also used skeletal tracking obtained from a 3D depth sensor to detect the direction of pointing in a similar manner (Srimal et al., 2017).

For speech recognition, some have used Google API (Maurtua et al., 2017) or the Microsoft Speech API (Gustavsson et al., 2017; Huang & Mutlu, 2016) for speech to text transformation. In the case of natural language, morphological analysis is performed to identify word morphemes and understand context. For example, Maurtua et al. (Maurtua et al., 2017) used FreeLing for morphosyntactic analysis while Nakata et al. [67] used the MOR and the POST program of CLAN. Nevertheless, Gustavsson et al. pointed out that relying on speech commands can be very problematic in the presence of background

noise and chatter (Gustavsson et al., 2017). Table 2.2 shows a summary of the key references related to body language and speech communication, and their advantages and limitations.

Table 2.2 Summary of key references in the Body Language and Speech subsection.

Description	Advantage	Limitation
A study on the use of natural implicit human gestures for controlling the robot	There exists a similarity between participant's natural body language which can be exploited for natural gestures control of robots	There are always exceptions and unpredictable motions performed by users. The translation between these natural movements and robot control is still primitive
A study of the method of creating a gesture dictionary that is as natural as possible and easily recognisable and differentiatable.	The suggested pipeline of choosing gestures based on how differentiatable they are from each other is effective and implementable in the foreseeable future	Training of workers is still recommended by the paper to make the execution of gestures more reliable. Moreover, it is still recommended to incorporate verbal commands or context awareness with gesture recognition to make the robot smarter.
A matrix represents relations between workpieces and actions that can be common ground for processing natural language instructions as well as sensory inputs and task planning	The technique provides solution to incomplete or imperfect language commands since the robot also performs task planning and observations to correct or confirm the	It is still unsure how the technique is extendable to more complex tasks
An interactive learning framework was developed that enables the robot to learn a task and deal with noisy, conflicting or incomplete inputs by asking the user intelligent questions	There was over 140% performance gain in noisy environment for a new situation when applying interactive learning model	The predicates are fixed and predetermined (almost rule-based). A challenge in the future would be to generate new predicates by interacting with the human or using deep learning
An algorithm was created that places objects on a rectangular table while merging gesture and verbal commands using fuzzy logic. They also account for obstacles, safety and robot reachability	Merging verbal and gesture commands produces more accurate results that when using only gestures or only verbal in some cases	This framework needs to be generalised and tweaked for tables of different shapes and objects of different sizes.
A voice and pointing command fusion engine was developed that merges information for commands together	There methodology is generic and extendable with making minimal or no coding modifications	The system doesn't perform yet in real-time due to its complexity
A GMM that models the robots hand position with respect to the position of a co-lifted table and the hand of the human partner	The robot predicts the human's motion and proactively caters for the change in position of the table	The prediction of the human's hand would become inaccurate if the motion was jerky
Utilising vision and force sensing to collaboratively keep a ball from falling off a table	Less cognitive load on the human since the robot takes a proactive role in balancing the ball. In case of disagreement with plan with human, the robot becomes more compliant	The work is task specific, and large modifications need to be performed to implement this on other tasks

Key References	Pohlt et al. 2017	Barattini et al.,2012	Cheng et al. 2018	She and Chai, 2017	Srimal et al. 2017	Maurtua et al. 2017	Sheng et al. 2015	Agravante et al. 2014

Attempting to use natural speech and gesture communication in an industrial environment is problematic since there is not enough industry-specific data to train models. Therefore, until natural speech and gesture understanding reaches a reliable level for industrial use, it is advisable to stick to a fixed set of verbal or non-verbal commands which are easier to recognise. However, issuing such commands should not be in each task iteration, since that would be mentally and physically tedious on the operator. But rather, that should be in special cases such as error handling. Moreover, such a communication scheme is especially suitable for Independent and Simultaneous scenarios in which the action sequence is more or less fixed and only occasional interference in required.

The intuitiveness of body language and speech communication is often traded with reliability. Therefore, exploring less human-like but more reliable communication modes, such as haptics and Graphical User Interfaces (GUIs), might be more suitable for industrial scenarios.

2.2.2. User Interfaces

Since cobots work closely with operators, cobots need to be equipped with intuitive UIs. These interfaces are used by operators to alter/create/customise cobot programs, whether off-line or on-line. The previous works related to cobot UIs and research challenges are discussed below. Table 2.3 shows a summary of key references in this subsection, their advantages and disadvantages.

User Interface Mediums

A UI is an essential differentiator of a cobot from traditional robots. Besides the user-interfaces being developed in research communities, several industrial solutions are already available. UIs are categorised such as:

- Cobot teaching pendant: Market cobots, such as Universal Robots (UR), ABB's YuMi and KUKA LBR iiwa, are labelled as intuitive and user-friendly due to their modular symbolic programming UIs. For example, the UR UI allows a user to specify way points and create arrayed motion patterns. The YuMi teaching interface is similar, with commands for both arms easily synchronised and parametrized. Teaching pendants are the easiest to utilise since they are built-in with the cobot. However, at a surface level, their capabilities are limited and do not enable human-awareness and action plan flexibility.
- Icon-based programming: A visual library of built-in functionalities can be utilised to create the program. For example, in MORPHA, the icons are connected to form a series of cobot commands. In LabVIEW, a data flow diagram is created in which values flow across the

icons and trigger actions on hardware. Although an icon-based program is easy to build (in small-scale programs), it is difficult to debug, maintain and alter. Therefore, they have not been popular in the manufacturing industry (Rossano et al., 2013). Moreover, these methods have not yet provided options to easily integrate the operator within the cobot's program so that the cobot is human-aware.

- CAD-based programming: Robot manufacturers and third parties have provided solutions such as V-REP, Visual Components and ABB's RobotStudio, in which performance can be validated and assessed. V-REP and Visual Components come with integrated human models that can also be programmed to help in validating the cobot's safety and collaborative functions around humans. In V-REP, a human can move according to a real-life actor through augmenting 3D sensor data, such as from Kinect v2, of a real-life human into the simulation. In Visual Components, the human can experience 3D simulation using VR which can be useful for training operators. However, since simulation has to match the real environment in order to achieve valid results, using CAD-based tools might be time consuming when changing work space design and reiterating the program, unless an automated method is devised to scan, map and build the environment.
- Task-based programming: This is the most popular research direction in intuitive cobot programming and will be further discussed in this subsection. The developed approach is based on a primitive-skill-task hierarchy (Schou et al., 2013): Primitives are cobot motion commands or sensory inputs values, such as open gripper and sense torque. Skills are object-oriented and achieve goals such as pick object and tighten screw. Tasks are a sequence of skills and achieve the over-all goal, which is the industrial scenario being implemented.

UI Design

Skills are the building blocks of task-based programming. They present a balance between specificity and abstraction, i.e. skills are general enough to be building blocks of a wide range of tasks while maintaining a level of abstraction understandable to humans. A skill structure was designed by Schou et al. (Schou et al., 2018). A skill transfers the environment from a state to another. Skills have preconditions that need to be checked before implementation. They also have post conditions that are checked to make sure that the skill is correctly implemented. Moreover, skills need to be parameterised depending on their input states, and continuous evaluation takes place during execution to ensure safety and right progress. Steinmetz et al. identified four key considerations for efficient skill design (Steinmetz & Weitschat, 2016):

- It is better to teach a parameter when needed so that an operator can assess the environment and choose the parameter accordingly.
- If the cobot fails to perform a task after parametrization, it should solicit the operator to edit the parameter.

- To avoid excessive parametrization, static knowledge about relationships between parameters can be utilised to reliably derive some parameters from others.
- Instead of being specified by shop-floor operators, some parameters should be set at defaults.

Moreover, the UI design should consider the four levels of users of UIs: Bystander, Modifier, Programmer and Integrator, each having different required competencies to operate the UI (Schmidbauer et al., 2020). The UI should be usable, understandable, intuitive, efficient, which are metrics defined by Marvel et al. (Marvel et al., 2020). Marvel et al. also provided design recommendations for UI designers to follow (Marvel et al., 2020).

UI Capabilities

For more sophisticated behaviour, researchers have worked on incorporating smarter motion generalisation, information display and cognitive abilities (including perception) in GUI architectures. Figure 2.2 shows a few examples.

- As shown in Figure 2.2 (a), Guerin et al. designed a GUI that allows the user to specify cobot capabilities and constraints (Guerin et al., 2014). Constraints include tool linear or planar path constraints. The user is also able to record tool affordances which include movement primitives (recorded paths). For example, using their GUI, the user is able to record a drilling action (straight constrained motion along the drill axis) and reproduce the action in novel drill locations.
- As shown in Figure 2.2 (b), Pedersen et al. represented a small set of skills needed in industrial cobots in a GUI (Pedersen et al., 2014). A set of high-level skills (e.g. pick-and-place) can be parametrized by a single input (object or location) through pointing gestures. The skill set supported is limited, but they elaborated on their work in Pedersen et al. (Pedersen et al., 2016).
- As shown in Figure 2.2 (c), Steinmetz et al. improved on the skill architecture and parametrization to support more complex skills, such as screwing (Steinmetz & Weitschat, 2016). Their work was formalised as a UI called RAZER and evaluated in (Steinmetz et al., 2018). RAZER allows an expert user to intuitively design new cobot skills and parametrize them. It presents these skills and parameter options to shop-floor operators for easy task-programming.
- As shown in Figure 2.2 (d), Schou et al. designed an interface that allows users to sequence skills and specify some pre-defined parameters (C. Schou et al., 2013). Locational parameters, e.g. having to do with the location of pick up, are specified using kinaesthetic teaching.

- Koch et al. incorporated the skill architecture in a software system named Skill Based System (SBS) that enables the creation of skills for complex tasks such as screwing and assembly (Koch et al., 2017).
- Paxton et al. designed the GUI for cobot programming, based on Robot Operating System (ROS), which is symbolic, modular and expandable (Paxton et al., 2017). Objects and agents (humans and cobots) are represented in a natural abstraction the human understands. These abstractions are used to generate Behaviour tree-based task planners using pre-defined actions. The operator can also specify way points for the cobot's path.

The aforementioned UIs provide intuitive solutions for programming a cobot for industrial tasks by workers with minimal programming experience. A flexible cobot behaviour obtained by the UI is a result of its use on-the-fly according to the operator's plans. In some cases in task-based programming, an expert designs the skill sequence and leaves some of the parametrization to be done by the operator on-line. This parametrisation is done by inputting values on the UI, by kinaesthetic teaching or by pointing gestures. The last two are only available for specifying locational parameters.

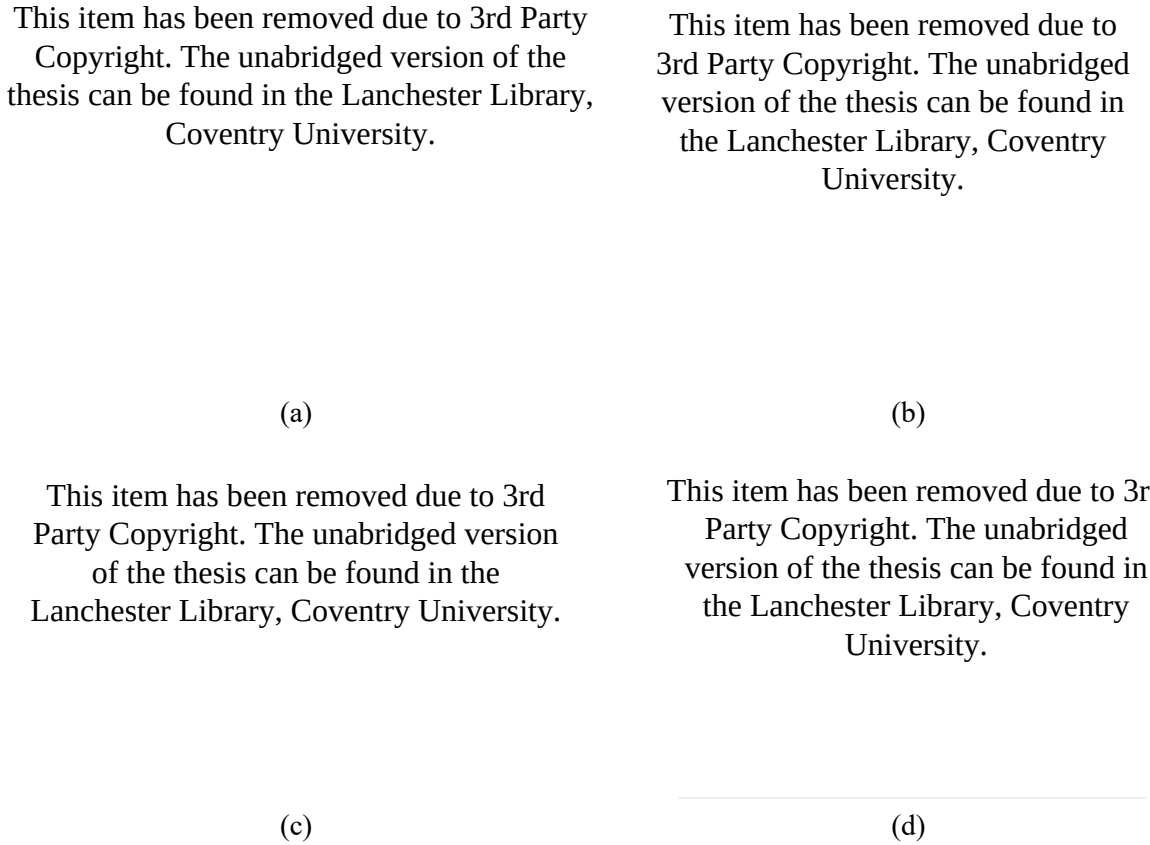


Figure 2.2 Examples of user-interfaces used to program cobots: (a) (Guerin et al., 2014)*, (b)* (Pedersen et al., 2014)*, (c)* (Steinmetz et al., 2018) *and (d)* (C. Schou et al., 2013)

A UI is an essential part of programming a cobot system whether by an expert or an operator, unlike the other technologies discussed in this chapter that can be scenario/task-specific and optional. Even

when other technologies are used to create/alter the cobot's program, a UI is still necessary to override any of them since it is the most reliable means of controlling the cobot.

2.2.3. Haptics and Force

Commercial cobots come with a built-in "compliant" mode, i.e. the cobot moves according to the forces the human exerts on its body. It can be considered, thus, that the human commands the cobot explicitly through touch and force. Researchers have extended on the default "compliant" mode to increase the cobot's intelligence and user-friendliness. That is, beyond detecting a force, understanding and reacting to it, researchers have worked on predicting user intent and negotiating plans. Table 2.3 shows a summary of key references mentioned in this subsection, their advantages and limitations.

Reactive Compliance: Understanding and Reacting to the Force

In reactive compliance, a cobot senses the forces exerted on its body and actively moves such that the forces are minimised. The challenge in reactive compliance is correctly mapping between the forces sensed by the cobot and the required motion to be done.

Most cobots are not equipped with tactile sensors on their bodies (the links between the joints), which makes it hard to localise and measure touch forces on the cobot's body. The human intent behind touching the cobot, i.e. trying to move it or stop it, becomes hard to understand. The cobot, for example, cannot identify the point at which contact with the human occurs and whether this contact is accidental or deliberate (Noohi et al., 2016). Magrini et al.'s work helps localise the forces being applied on the cobot (Magrini et al., 2015). The method is based on the integrated use of model-based residual signals that detect the occurrence of collisions and of one or more external RGB-D sensors to approximately localise the contact point on the surface of the robot links. That allows the cobot to respond to the contact force as desired or regulate it. Kouris et al. differentiated between collision and cooperation contact in a computationally efficient manner by thresholding the Fourier transform of the applied force/torque (Kouris et al., 2018). Gaz et al. differentiate between forces applied due to a polishing task and forces applied to move the cobot body by using a model-based approach (Gaz et al., 2018). This allows the human to smoothly and safely switch from moving the cobot compliantly and performing the polishing task.

Forces can also be difficult to interpret when they are exerted on the object a cobot is holding rather than directly on its body. The force that a human exerts on an object can signify an intent of motion in different directions. Wojtara et al. devised algorithms that differentiate between rotation and translation motion intent in a collaborative object-positioning scenario (Wojtara et al., 2009). The first algorithm relies on degree-of-freedom (DOF) switching where the human explicitly specifies his/her intent (rotation or translation) and the cobot acts such that the right DOF are varied or fixed. In another algorithm, "Partner-that-follows", Wojtara et al. interpreted force as translation and torque as rotation intent above the human's axis. The results section is used to assess the different algorithms proposed

and compare them. All the algorithms present a reliable way of controlling a cobot for co-manipulation. However, there is no evolutionary element that allows the cobot to adjust with time to its human partner and the algorithm is very human-led. However, the work of Wojtara et al. is the closest to industrial feasibility due to its reliability and predictability.

As the number of potential directions of motion increase, it becomes difficult to manually toggle between them. Dumora et al. used learning algorithms to map from sensed forces to required direction of motion (Dumora et al., 2013). A Naive Bayes classifier is trained with the input vector of static forces on hand-held nob, and the output being the intent of direction of motion. The cobot then provides compliance in the intended direction.

Reactive compliance produces reliable results, which increases the level of trust the operator has in the cobot. However, since the cobot only moves under the influence of the operator, he still carries a mental and physical burden. To decrease this burden, the deeper understanding of the human intent and goal are needed in order to take a more proactive role.

Proactive Compliance: Supporting the Human's Intent

Researchers have worked on deepening the cobot's understanding of the human's exerted forces to behave in a more proactive manner. Once the cobot understands the direction the operator wants to move the cobot in, the cobot exerts torque that supports the operator's intent. The challenge in this degree of compliance is the accuracy of the inferences made from the exerted forces and the validity of their utilisation. For example, Li and Ge used force to estimate desired target positions (Li & Ge, 2014). Estimating the human's desired target position decreases the amount of force he/she should exert as the cobot takes a more proactive role. This is achieved by integrating the predicted motion intent of the human into an impedance controller. The algorithm, however, assumes that the human's intended motion path is smooth and continuous. Therefore, a sharp change in intent results in higher needed torque and more time than a regular impedance controller. Lichiardopol et al. worked on decreasing the physical load on the human while assigning the cognitive responsibility to him/her (Lichiardopol et al., 2009), i.e. the human guides the path of the co-manipulation task with minimal exerted force. They assumed that the object's weight is unknown and potentially time-varying. Therefore, the algorithm estimates the force the human is applying based on the cobot's control torque and the position change. Then, the cobot amplifies its torque to decrease the estimated human exerted force. Moreover, the mentioned estimation and amplification steps happen in periodic cycles to cater for changing object weight.

Incorporating more intelligence and inference/prediction abilities in cobot programs decreases the physical and mental load on the operator. However, it also increases the probability of failure and unexpected cobot motions. Therefore, a more clear-cut between autonomy and reactive compliance would potentially avoid uncertainty and relieve the human of burden at the same time.

Mixed-Initiative Compliance: Balancing Between the Cobot's and Human's Plans

A cobot has a goal path or position and acts autonomously to fulfil the goal. When an operator exerts force on the cobot, the system assesses how autonomous it should be as opposed to compliant. In some cases, the switch between the two modes is clear-cut, while in other cases the trade-off is smooth. The trade-off can be done by adjusting the stiffness values in impedance control or by weighting the autonomous and compliant components to achieve a combined result.

Table 2.3 Summary of key references in the User-Interface and Haptics subsections.

Advantage	**Limitation**
The UI enables motion-level and task-level programming in an intuitive way	The UI misses the intelligence needed to perform flexible tasks
Capabilities can be adjusted as to be performed on different locations with physical constraint	According to the paper, a limitation is that the UI still doesn’t handle failures
Skills account for pre-conditions and post-conditions which means that there is potential for task planning to be embedded in the system	The performance of the robot programmed using the UI was significantly slower that when it was explicitly programmed to perform a certain task, as mentioned in the paper
The UI has been praised for its frontend by the participants in the experiment of trying it and comparing it to other UIs	The programming possibility to include conditionals by incorporating pre- and post-conditions, needs to be included.
The tool includes perception capabilities that enable the generalisation of skills as well as a SmartMove feature that enable the automatic action selection based on pre-conditions	The representation of tasks as a behaviour tree is not very intuitive to understand
The classifier was able to classify the motions with a very low error	The classifications are discrete which is not true in real-life where motion might need to be smoother (weight need to be given to different motions given the force as opposed to just choosing one motion to apply)
The method helps the human move the robot to a desired point with less time and less torque than using impedance control	It is assumed that the target position is fixed with respect to the robot. And that the path is smooth. A quick change in the target position or the human’s intention is not considered

Key References	Description
Schou et al., 2018	User interface is created that allows users to create, parameterise and sequence skills to program cobots
Guerin et al. 2014	A user interface where capabilities are designed for robots and then assembled to form tasks
Pedersen et al. 2016	A skill-based programming UI was designed for cobots
Steinmetz et al. 2018	An interface is created that allows the creation, parametrisation, and sequencing of skills
Paxton et al. 2017	A powerful ROS-based task planning UI is created
Dumora et al. 2013	It used Naïve Bayes to classify intended directions of motion (rotation/translation with respect to x/y axes) given a force applied on a robot's end effector
Li and Ge 2014	A Neural network is trained to estimate the human's position intention based on the exerted force, in order to make the robot act actively towards the intended position

For example, consider the case where a cobot knows a predefined path while the human's intended path does not fully align with it. In such a case, the cobot must know when to favour its own path and when to switch to being compliant to the human, i.e., when to use its control torque input and when to use the human's applied control force. Li et al. solved this by adjusting the weights are adjusted to minimise the difference between the applied human force and the "optimal human force" given the current motion direction (Li et al., 2015). Briefly, when the applied human force matches the pseudo-force applied in the direction of motion, the cobot relies more on its own controllers to maintain the direction of motion. However, when the human force changes and is not aligned with the current direction of motion, the cobot becomes compliant and relies more on the forces to move rather than on its torques. This is similar to an impedance controller with autonomously varying damping and stiffness. The proposed algorithm creates smoother compliant motion while relieving the human from the continuous needed effort to push the cobot.

However, in an industrial scenario, an operator will perform similar compliant motions for numerous times. The algorithm can also incorporate a learning element that compiles observed motion patterns and seeks to reproduce them while also being flexible to deviate from learned paths according to the human's current plans. An example scenario is co-moving a heavy object from Zone A and performing a precise positioning in Zone B. When approaching the object of an uncertain position, the human would naturally lead the cobot since he/she is equipped with better perception skills that allow more precise positioning. Similarly, the human would tend to take the lead when precisely positioning the object in Zone B. However, moving between zones can be done by the cobot after being led a few times by the human. Rozo et al. implemented a "learning from demonstration" algorithm that learns cobot stiffness

from a set of kinaesthetic demonstrations (Rozo et al., 2016). The demonstrations are parametrized according to the position of objects, obstacles and the human and represented as a Gaussian Mixture Model (GMM). This algorithm proved more robust against unobserved positions and varying forces exerted by the human, as opposed to control algorithms with fixed stiffness.

Agravante et al. combined reactive and proactive behaviours by relying on both haptic and visual inputs (Agravante et al., 2014). The task handled is co-lifting a table while keeping a ball on it. The impedance controller which relies on haptic information, i.e. the forces sensed from the human, provides compliant behaviour in all 6 DOF. The vision controller only controls 2 DOF (z and φ_x) such that the ball is "attracted" to the centre of the table. In the case of intent conflict, the impedance parameters are adjusted such that the cobot becomes more compliant and less stiff. Sheng et al. also merged proactive and reactive behaviours (Sheng et al., 2015). In the problem, a cobot has to grasp a table side (gross motion) and then co-lift it with a human such that it remains horizontal (fine motion). The cobot learns how to approach and grasp the table using LfD. When the cobot successfully holds the table, is uses an RL-based reactive controller to keep the table horizontal with the human. A proactive controller predicts the human's position (equal to the cobot's required action) in future time steps using a Kalman filter and assuming constant acceleration. A behaviour gain controller then merges the suggested next-step action from the reactive and proactive controllers. The integrated algorithm combining the reactive and proactive performed better than just reactive algorithms. In conclusion, the trade-off degree of compliance provides a balance between minimising mental and physical load on the operator while also yielding predictable controllable cobot actions.

Aside from the challenges of designing the communication channel between a human and a cobot, Unhelkar et al. tackled the issues related to decision making in communication (Unhelkar et al., 2017). That includes the question of if and when to communicate, which relates to the cost and benefit of communication and the estimation of the human's mental state. They present open questions of how to quantise the cost of communication and its benefit to decide whether/how communication should be used. Since communication required explicit involvement from the operator, it can be mentally and physically tiring in repetitive long industrial tasks. Incorporating flexible autonomy through optimisation or learning, which will be discussed below, is an alternative.

2.3. Optimisation

Optimality is a primary goal during industrial design processes (product, process and production line design) since it ultimately yields a "maximum" profit. The main challenge in HRC scenarios is to optimise around the human, i.e. modelling and incorporating the human in the cost function. This subsection reviews the works done on optimising different aspects to yield optimal and semi-optimal cobot action in different industrial HRC scenarios.

2.3.1. Modelling different human states

Usually in repetitive non-collaborative industrial scenarios, processes are optimised with regard to minimising time, waste and maximising quality and profit. Obtained parameters from the optimisation process are incorporated in programs and control algorithms that dictate cobot actions. In HRC scenarios, however, the human is a central part of the cobot's surrounding, affecting its performance. Modelling the human is a challenge due to the high number of factors and their unpredictable variability. Researchers have attempted to quantify or estimate human factors such as trust, physical load and mental state using observable and measurable states.

Since ergonomics is a main driver of implementing HRC, much has revolved around producing cobot behaviour that maximises humans' physical comfort and health. Modelled human factors related to the human's physical state include:

- Static ergonomic posture according to REBA: Busch et al. optimised cobot pose during handover to achieve human ergonomic posture (Busch, Maeda, et al., 2017). They account for left/right handedness and avoid intimate body parts, all while keeping the human body in a safe comfortable posture according to the Rapid Entire Body Assessment (REBA).
- Muscle fatigue: Peternel et al. measured human muscle fatigue in order to adjust cobot's behaviours such that it handles more physical load and the human takes a more supervisory role (Peternel et al., 2016). The cobot does not take on all the physically-loaded tasks from the start since it needs to learn them from the human first. Hu and Chen estimated dynamic human fatigue (varying with time as the human works more) per assembly action and accordingly distributes tasks between operator and cobot (B. Hu & Chen, 2017).
- Human joint torques: A key work in optimising co-manipulation for ergonomics is done by Peternel et al. (Peternel et al., 2017), a follow-up work of (Kim et al., 2018). In both, the human's pose is optimised during co-manipulation or handover task, such that the torque on the human joints are minimised. Table .2 highlights the main differences between the two works. An extension of these works would be to merge the benefits of both by mathematically remodelling the problem.

Table 2.4 Comparison between the works of Kim et al. (Kim et al., 2018) *and Peternel et al.* (Peternel et al., 2017)

Kim et al. (Kim et al., 2018)	**Peternel et al.** (Peternel et al., 2017)
Only semi-static forces applied on the human were accounted for.	Forces obtained from dynamic motion were also accounted for.
The centre of pressure (CoP) was measured using a pressure sensor plate the human stands on.	The CoP was estimated using the weight of held object.

The forces need to be applied on the human before they can be minimised.	The forces are predicted and optimised before applying them on the human.
Can be used in co-manipulation scenarios.	Can only be used in handover scenarios in which the human carries the entire load.
Any force applied on the human by the object or the cobot can be accounted for.	Only vertical forces applied by weights of the object held are accounted for.

Optimising for ergonomics also includes accounting for the human's mental model/state, including:

- Human knowledge of task: A mental model includes the human's knowledge of a task which, if known, helps the cobot assist only where and when needed. Milliez et al. designed a task planner that enables a cobot to decide when to instruct the human through a task, when to do the task itself and when to monitor the human's performance (Milliez et al., 2016). Their planner accounts for the human's expertise which is based on successful task attempts. Such a planner is useful in industrial situations since the cobot can know how much interference in the task is required depending on operator experience. Devin and Alami designed a framework that estimates the human's mental states, i.e. the human's knowledge of the environment, plans, progress and goal, and triggers the cobot to only communicate with new information to the human when needed (Devin & Alami, 2016). In tasks where several goals are possible, Zhu et al. estimated the goal belief of the human and optimised their action sequence such that the wrong goal of the highest probability is eliminated (Zhu et al., 2017). Several other works studied the preference of humans for proactive (perform sub-tasks autonomously) versus reactive (perform tasks when triggered or asked for help) cobots (Baraglia et al., 2016; Schulz et al., 2017).
- Trust in cobot: The mental state also includes a human's emotional state, i.e. stress/trust level. Sadrfaridpour and Wang controlled the cobot joint velocity while accounting for the estimated human trust level (Sadrfaridpour & Wang, 2018). The trust level is estimated based on the progress of the human along his path while working alongside the cobot. For instance, a human is moving unusually slowly is being wary and careful and thus assigned a low trust value. The trust value is then fed into a non-linear model predictive controller (NMPC) to obtain control inputs. Compared to a controller that only aims at synchronising the human and cobot’s motions, the trust-integrated NMPC resulted in a higher trust level and less perceived workload for the co-worker human. In scenarios where the human and the cobot co-lift a work piece, the human performs better with a cobot that moves along a path in a biological velocity pattern rather than a fixed velocity (Maurice et al., 2018). Huang et al. created an algorithm that slows down its motion to match the human's pace and task progress (Huang et al., 2015). It shows that this is preferable as opposed to a cobot that

executed its motion at a fixed pace and remains idle until the human catches up. Research was also done to produce legible cobot behaviour, which helps the human anticipate the cobot's intentions and increases the trust level (Bodden et al., 2016; Busch et al., 2017).

Accounting for the human's mental model/state might be regarded as an overshoot by industrial parties. Moreover, since the mental state/model is estimated rather than measured, this presents unnecessary uncertainties in manufacturing processes, especially when a task-level decision is being made. However, during a motion-level decisions, giving indications of the cobot's intent is desirable since it boosts the human's comfort and trust in the cobot. This can be done by moving in a legible path (Busch, Grizou, et al., 2017), by communicating through light signals (Tang et al., 2019), by displaying facial expressions (Reyes et al., 2019).

2.3.2 Balancing between Human and Task Benefits

However, as mentioned earlier, the goal of optimisation from the industry's standpoint is not merely to ensure better comfort for the human operator. Task parameters should be selected to minimise loss and time. Besides accommodating the human operator, the industry is interested in optimising towards task efficiency, i.e. improving product quality and decreasing production time (which can be estimated (Pellegrinelli et al., 2016)). Faber et al. used CAD information to optimise assembly sequence to achieve low mental and physical load on the human, and minimise the number of cobot tool switches and human-cobot switches (Faber et al., 2017). Johannsmeier and Haddadin distributed assembly sub-tasks between a human and a cobot as to minimise workload or energy consumption per subtask (Johannsmeier & Haddadin, 2017). Hawkins et al. predicted human actions probabilistically to optimally enlist cobot help and minimise wait time (Hawkins et al., 2015). This probabilistic prediction is based on observations of the human's previous actions and observation reliability and trust.

Table 2.5 Summary of key references in the Optimisation section.

Limitation	The algorithm only accounts for static poses. Markers are used to perform visual detection	The experiment was performed on one subject so it is unclear if and how it will work on general population	It works only for vertical forces such as in handover tasks	Force needs to be applied on human before they are minimised	It is assumed that the human can always understand the robot when communication occurs	The modelling in the algorithm is task specific which makes the framework costly to implement for more tasks	It is assumed that the robot is fully autonomous, and that the human cannot interfere with the robots motion, which is often untrue

Key Reference	Description	Advantage
Busch, Maeda et al. 2017	The robot's posture is optimised after accounting for task constraints and the REBA human posture equations	The optimised pose is preferred and safer for the human's posture. The algorithm is generic and works for any task
Peternel et al. 2016	Human muscle fatigue is measured and adjusted the robots behaviour in order to minimise it	Once the robot takes over most of the physical load, the human can still choose to intervene if needed
Peternel et al. 2017	Estimate the Cop based on weight of held object and adjusts robot to minimise force	Dynamic forces can also be accounted for
Kim et al. 2018	the Cop is measured using a floor pressure sensor plate and adjusts robot to minimise the force	It can be used with any manipulation task regardless of direction of force
Devin and Alami 2016	A cobot was created that estimates the human's mental state and issues instructions to the human when s/he needs to be corrected or triggered	The amount of information shared by the robot to the human is significantly less that other systems that do not account for mental models and share any new information or change in plans
Zhu et al. 2017	The algorithm estimates the goal belief of human and eliminates robot actions that lead to the goal with the least probability.	Users agreed that the robot was more efficient, responsive and helpful that if the robot was just performing the more efficient" step towards the goal without accounting for the human's goal belief
Maurice et al. 2018	A biological velocity pattern was modelled for robots to be more natural	The human performed better and more comfortably when the robot was moving biologically

The advantage of using optimisation is that it yields optimal cobot behaviours. However, an optimal behaviour may sometimes conflict with the human's plans or preferences. Game theory enables cooperation between agents (humans and cobots) such that mutual benefit is realised, which is why it has been utilised to produce rational cobot behaviour accommodating human interest. It combines intelligence and rationality in pursuing one's goal while considering the plans and benefit of other agents. Gabler et al. modelled a human-cobot close proximity pick-and-place problem as a two-player game and uses a Nash Equilibrium to solve a cost function that accounts for travel effort, object reachability, and object preference as well as collision risk (Gabler et al., 2017). Li et al. used game theory to switch the cobot role from leading (assuming a predetermined path) to compliant (deflecting from the planned path) in a co-manipulation task, based on forces exerted by the human (Li et al., 2015). Gombolay et al. extended a dynamic scheduling algorithm, Tercio, to accommodate human preference, workload and situational awareness (Gombolay et al., 2017), applied in object fetching.

Banziger et al. used a simulation tool in order to optimise task allocation in several collaborative tasks (Bänziger et al., 2018). The use of a simulation tool allows to calculate several human and task

parameters such as ergonomics and production time. It also allows to measure these parameters in multiple task distributions, which enables finding the optimal task allocation. On the other hand, Yu et al. created a task scheduling algorithm for assembly tasks by considering assembly moves analogous to the AlphaGo Zero game (Yu et al., 2020). They used a similar algorithm based on reinforcement learning and a CNN to optimise the next move choice after designed "game rules" for an assembly "chessboard".

Table 2.5 shows a summary of the key references mentioned in this subsection, their advantages, and limitations. The prevalent limitation with optimisation is that optimisation models are manually designed. Human states, although capturable in a model, remain relatively simple. This gives rise to learning algorithms that allow the modelling of actions and human states automatically learnt from data.

2.4. Learning

Humans learn new tasks by observing them being done, by trying to do them and by asking questions and receiving feedback on performance. A human teacher serves to demonstrate a task, answer questions and provide feedback, all of which do not require programming skills. Researchers have attempted to enable a learner-teacher relationship between the cobot and the operator due to its naturalness and wide potential. In HRC, it is advisable to equip the cobot with learning capabilities since the operator might have to expand its skill set due to unforeseen working circumstances. Moreover, learning provides a balance between allowing the operator to make decisions first, then relieving him/her of the mental load as the cobot learns to operate autonomously.

2.4.1. Learning from Demonstration

LfD is a very popular programming method in HRC due to its apparent intuitiveness and convenience. Research focus has revolved around capturing demonstrations reliably and easily, and encoding accurate state-action information to reproduce the task robustly in a new environment.

Table 2.6 Advantages and disadvantages of different demonstration recording techniques

Method	**Advantages**	**Disadvantages**
Human demonstration, e.g. (Lafleche et al., 2019): the human records him/herself doing the task. The cobot needs to extract object-goal relations or other useful information from the observed demo or relevant human joint paths to replicate.	Easiest for the human to perform	Not applicable to scenarios only done by the cobot (lifting heavy objects), detecting the human pose might be inaccurate, mapping the human pose to cobot pose is a challenge (correspondence problem)

Kinaesthetic teaching: the human holds the cobot and moves it as required by the task. The cobot is in compliant mode.	Straight-forward to perform and setup since compliant mode is a built-in feature for market cobots	Is difficult to perform with bulky or heavy cobots (e.g. UR10), may generate a shaky paths, not suitable for high precision tasks, not suitable when there are spatial constraints around the cobot
Teleoperation, e.g. (Fischer et al., 2016): the cobot is controlled remotely using an external device and the path generated is recorded as the task demonstration.	Might be intuitive and fun for some operators, can yield very smooth and precise paths depending on the device used, its sensitivity and calibration	Setting up and calibrating the device is a lengthy process prior recording the demonstrations, causes discomfort for some operators who find cobot motion "unpredictable"
User-Interface: the cobot is controlled using teaching pendent, whether by joint movement or end-effector movement.	Some movements are easier such as gripper rotation, predictive and consistent	Not instinctive, takes a long time, tedious, reaches a lot of singularities and needs resetting often

Recording Demonstrations

Recording or displaying demonstrations can be done in several ways, each with advantages and disadvantages (Table 2.6). Aside from human demonstrations, kinaesthetic teaching is the fastest way of recording a demonstration (Fischer et al., 2016) and is generally preferred by users (Akgun & Subramaman, 2011). Teleoperation is highly dependent on the device used and it would yield comparable results to kinaesthetic teaching depending on the design (Fischer et al., 2016).

Other solutions might fall in a grey area between the aforementioned methods. When teaching by kinaesthetic demonstration, the human is often in an uncomfortable position moving the cobot and teaching the cobot's path causing unnecessary jerks. Also, only the trajectory knowledge is transferred and not stiffness information. Therefore, Yang et al. presented a hand bracket interface that allows a human to naturally and comfortably move a cobot (Yang et al., 2017). Moreover, electromyography (EMG) signals are measured from the human's muscles which are transferred to the cobot as stiffness information for the impedance control and as open/close commands for the gripper state control. However, with their current hardware design, the cobot must have two arms. Different methods can also be used concurrently in the same system, to learn different aspects of the task (Y. Gu et al., 2018). This depends on the demonstration encoding requirements and the pros and cons of the different methods.

Encoding Information from Demonstrations

LfD algorithms differ in what information they encode from the demonstrations, making it difficult to design one that caters for all expected variability in the environment and capture all requirements. Encodings can be motion-level or task-level. Motion-level encodings include:

- Motion with variable force: Kramberger et al. used Dynamic Motion Primitives to learn the trajectory of scooping small parts from a container (Kramberger et al., 2020). They switched force controls which are learnt during the demonstration in order to ensure the successful motions when in contact with the container and to avoid getting jammed by the parts.
- Motion with respect to obstacles: Ghalamzan and Ragaglia encoded obstacle presence from demonstrations, so that the reproduced cobot actions could avoid moving obstacles to reach a target location (Ghalamzan E. & Ragaglia, 2018).
- Motion of two agents (humans or cobots) with respect to each other: Vogt et al. encoded correlation-based interaction meshes from one human-human demonstration where one human led whereas the other followed. Then, they reproduced cobot motion that matches the human follower's pose with respect to the human leader while avoiding the correspondence problem (Vogt et al., 2017).
- Constrained position and path with respect to objects and tools: Perez-D'Arpino and Shah encoded required postural (relative to work pieces) and path constraints for multi-step tasks (Perez-D'Arpino & Shah, 2017).
- Compliance level (Stiffness) as a function of path: In a co-manipulation task, Rozo et al. encoded compliance level (stiffness) given force and position inputs and could therefore reproduce co-manipulation behaviour with the right stiffness (Rozo et al., 2016).
- Path dependent on position of landmarks: In task-parametrised LfD (Calinon, 2016), the path of the cobot is encoded with respect to multiple landmarks as opposed to one. The importance/relevance of these landmarks to the path is automatically calculated from the variance of the path between multiple demonstrations. Task parametrised LfD has been used to generate trajectories for assembly tasks (Duque et al., 2019).

Task-level encodings include:

- Encoding action preconditions and effects: Liang et al. encoded task-level information from kinaesthetic demonstrations (Liang et al., 2017). The task preconditions and effects were extracted and used to create action models. During run-time, pre-conditions were identified by the cobot and the suitable action model chosen to create the desired effect.
- Encoding action sequence: Maeda et al. used demonstrations to encode different sequences of human-robot actions to accomplish the same task (Maeda et al., 2016). During runtime, a lookup table is used to identify the most likely sequence followed according to the human's observed actions to predict and provide the complimentary cobot actions. Also, Hamabe et

al. also generated the task finite state machine (FSM) from a set of demonstrations which was used during runtime to identify the required supportive action (Hamabe et al., 2015).

Other algorithms encode both motion and task-level information. For example, Gu et al. created the Portable Assembly Demonstration (PAD) system that learns task-level and motion-level skills from human demonstrations and kinaesthetic teaching, respectively (Y. Gu et al., 2018). The system detects parts and tools and automatically recognises assembly states, actions and parts/tools involved, after observing the human demonstration. Kinaesthetic teaching was used to learn primitive actions that enabled these skills. Their system is robust to occlusions and environment changes and is able to handle complex assembly tasks such as screwing, wrenching and hammering.

Time Aligning Demonstrations

Time alignment is important when demonstrations and execution are not guaranteed to run exactly the same rate. Time alignment, such as dynamic time warping (DTW) (Vogt et al., 2017), is used to temporally match the demonstration with the sequence of states observed so far. However, problems might arise when the performance velocity differs drastically from demonstration to execution. Therefore, Maeda et al. rely on phase estimation instead which accommodates different velocities of human motion (Maeda et al., 2017).

Expanding the Demonstration Set

Another challenge in LfD is how to generate enough demonstrations showing the right variability, as doing so is time consuming. Moreover, how should one make sure that demonstrations are being generated usefully, and are not being redundant? Forbes et al. relied on a seed demonstration and then solicited a crowd to edit the demonstration using a GUI for all the scenarios this demonstration would fail in (Forbes et al., 2014). Luo et al. used on-line learning to expand the library of demonstrations when required (Luo et al., 2018). Arm reaching motions are encoded as a GMM library, and during run-time, the partial trajectory is identified in the GMM. A GMR is used to predict the rest of it, allowing the prediction of reaching target. When a new reaching motion that does not resemble the existing GMM is recognised, the GMM library grows. Mohan and Bhat presented a growing, multi-modal memory framework that encodes diverse experiences of the cobot in HRC settings (Mohan & Bhat, 2018). It recalls past experiences in present context to plan future action. Their framework contains a perception system that stores object information as well as an action system that stores motion plans. The two systems interact together and with the Episodic Memory system that encodes experiences, infers goals and plans.

LfD is a special case of supervised learning, in which a set of truths are given and learned from. Supervised learning can also be used to map between states and required actions. For example, in (Dumora et al., 2013), a Naive Bayes classifier was trained to output the direction of cobot compliant motion when given a vector of static human-applied forces on end effector.

2.4.2. Reinforcement Learning

RL has been used to teach intelligent robots skills such as grasping objects of irregular shapes (S. Gu et al., 2017) and a UAV avoiding obstacles (Singla et al., 2021). Robots are left for extended periods of time training the RL policy. In some cases, (Levine et al., 2018), for grasping tasks, several robots are trained at the same time and knowledge is shared. In an industrial situation, such training time and resources might not be available. To combat the time limitation, demonstrations are incorporated in RL in order to facilitate and guide the learning process. Rajeswaran et al. used a human demonstration and RL to teach a robotic hand dexterous tasks, such as nail hammering (Rajeswaran et al., 2018). The demonstrations are used to initialise the RL policy, which facilitates convergence towards optimality. The demonstrations are also augmented in the loss function so that the converged policy maintains a similarity to them. In (Hangl, Dunjko et al., 2017), the cobot uses active learning to expand the applicability of a given basic behaviour to convert state A to state B, i.e. perform a certain task. In other words, given state C, the cobot autonomously explored a set of actions that would change it to state A so that the basic behaviour can be applied to convert to state B. Moreover, the cobot autonomously finds suitable perceptual actions that capture useful information about the environment given the task at hand. Their method was also integrated within a GUI that allows the user to easily program the initial basic behaviour (Hangl, Mennel et al., 2017).

In HRC cases where a human is part of the cobot's environment (observed states) specifically, standard (or “vanilla”) RL is used since the operator cannot reasonably complete the numerous learning iterations with the cobot. In (Nikolaidis et al., 2015), cross-training (in which the human and the cobot switch roles during the training process to facilitate learning) is used to learn the reward function for the cobot's collaborative actions. Gu et al. used RL to collaboratively balance a table with a human. The reward, rather than being human feedback, is the change of slope of the table (Y. Gu et al., 2011). Sheng et al. added to that a proactive element to predict the human's intention and varies the table's slope accordingly (Sheng et al., 2015).

Table 2.7 shows a summary of the key references mentioned in this subsection, their advantages and limitations. Learning-related program features provide autonomy, while enabling the operator to intuitively program the cobot via learning data. The cobot does not need continuous commanding from the operator yet behaves while showing awareness to the operator's presence and actions. Moreover, the operator can choose to alter the cobot's program by providing new training data, such as new demonstrations for the LfD algorithms. However, as aforementioned, since such algorithms are sensitive to the quality of data, not all data provided by an operator can yield desirable results. Moreover, since policies generated by such algorithms are usually non-deterministic and probabilistic, unexpected outlier results might occasionally be encountered.

2.5. Task Parametrised Gaussian Mixture Models

LfD is the most promising way of teaching a cobot, since it does not require a large set of data to train, can capture a wide range of task dependencies (depending on the chosen LfD algorithm) and is relatively intuitive for operators to perform. Although the process of using LfD is intuitive, operators are still encouraged to understand the theory behind it, since there are many decisions that need to be taken by them that require knowledge of how LfD works.

Task parametrised LfD is effective at capturing dependencies between different states in the environment, such as positions of objects and humans, and producing a cobot action accordingly. For Independent and Simultaneous scenarios, if objects have predetermined positions, then it is advisable to program the cobot by specifying fixed key points using built-in options in the cobot's teaching pendant. If positions of objects vary, TP-LfD can cater for this variance provided that the cobot is able to detect the positions of these objects. TP-LfD can even cater for position of the human's hand with respect to these objects.

Table 2.7 Summary of key references in the Learning section.

Advantage	Limitation
Smoother paths are generated than when doing kinaesthetic teaching	.Expensive custom-made hardware is needed
The setup process is easy and saves time compared to manual programming	The modelling in the paper is task specific
The task is recorded between 2 humans which makes the recording intuitive and most efficient	The algorithm has to solve the correspondence problem and inverse kinematics need to be calculated which is costly
allows non-experts to teach robots constraint tasks	The algorithm doesn't cover articulated constraints, such as opening a door, which is common in factories
The algorithm required human demonstrations which are easier to provide	The algorithm relies on custom vision algorithm to detect the different pieces which makes it hard to use it for another task
The users mostly believed that the interface was understandable and intelligent	A lot of the users were confused with some logic statements such as conditionals
The human's waiting time is significantly reduced since the robot always estimates what the human needs and fetches it before the human asks for it	If a step is not present in the look-up table, the robot tries to find the closest one to it which might be incorrect and would lead to frustrating interactions
The system is robust to occlusion and environmental changes	The fact that a hardware is involved can make the easy deployment of this system harder
The effort needed to record a new demonstration is reduced and the demonstrations collected from the crowd are ensured to resemble the expert's	The crowd-sourced demonstrations cannot be automatically tested and the crowd involved in the study was local co-workers instead of random people
The learning algorithm performed well and seldom made mistakes differentiating between the intentions	The method requires time consuming extra demonstrations for learning new tasks
The algorithm achieves high results with a simple sensory system (a monocular camera)	A large dataset is required for training which makes retraining hard when new objects are introduced to the production line
The algorithm works for robots of very high degrees of freedom such as 24-DoF five fingered hand	Since the training occurs in simulation, it is difficult to include a human in the task

Key References	Description
Yang et al. 2017	A bracelet is designed that enables the human to move the robots arm more intuitively as well as learn required robot stiffness from EMG hand signals
Kramberger et al. 2020	DMPs are used to learn force control for scooping small parts from a container
Vogt et al 2017	Interaction meshes are encoded between two agents to produce collaborative tasks
Perez d' Arpino and Shah 2017	Geometric constraints are learnt with respect to objects from demonstrations
Duque et al. 2019	TPGMM and Petri nets were used to learn the assembly of a wooden car
Liang et al. 2017	Symbolic language is used to enable non-programmers to program an assembly task
Maeda et al. 2016	A look-up table for tasks that the robot refers to during run-time is created to always cater for the task step that the human is performing
Y. Gu et al. 2018	A rotating platform learns the models of objects to be detected in demonstrations in TP-GMM learning
Forbes et al. 2014	A system was created where when a demonstration fails in a certain scenario, a crowd member adjusts the main demonstration to make it work
Dumora et al. 2013	The robot responds to haptic cues from the human to understand his/her intention and respond to it
Levine et al. 2018	A CNN is trained to map between an image of cluttered objects and a motor command signal to a probability of success of grasp
Rajeswaran et al. 2018	A DRL algorithm is used to train a robot hand to grasp objects given a few demonstrations

2.5.1. Training

Assume M demonstrations consisting of T data points each. Each data point $\xi_{m,t}$ is D dimensional, observed from a global frame of reference. The point dimensions depend on the task but popular options for dimensions are spatial coordinates, velocity, time step, and force values. Also, assume P task parameters, each being a frame of reference in space, defined by $\{b_{p,m}, A_{p,m}\}$, where $p \in \{1 \dots P\}$ and $m \in \{1 \dots M\}$. $b_{p,m}$ is the position vector and $A_{p,m}$ is the rotation matrix of task parameter p in demonstration m.

The D-dimensional points ξ_t for all demonstrations are transformed from the global frame of reference to local coordinate frame of each frame of reference p, such that

$$X_t^{p,m} = A_{p,m}\left(\xi_t - b_{p,m}\right) \tag{2.1}$$

$$X_t^{p} = \left[X_t^{p,1}, \dots, X_t^{p,M}\right] \tag{2.2}$$

The data points from all demonstrations are concatenated together in one vector to form X_t^p. A TP-GMM is trained to model all X_t^p for $t \in \{1 \dots T\}$. For each parameter p, the GMM consists of K components that probabilistically cluster the demonstrations path points observed from said parameter. Each component is defined by $\{\mu_{p,k}, \Sigma_{p,k}\}$ where $\mu_{p,k}$ and $\Sigma_{p,k}$ are the centre and the covariance matrix of the k^{th} Gaussian for frame p. Moreover, a mixing coefficient π_k is defined for each Gaussian $k \in \{1 \dots K\}$.

The training of the TP-GMM is done using expectation-maximization (EM) algorithm to iteratively update the Gaussian component parameters until convergence. In the expectation step, the likelihood a point X_t^p belongs to each Gaussian component $\{\mu_{p,k},\ \Sigma_{p,k}\ \}$ in the GMM is calculated. In the maximisation step, the Gaussian components are recomputed to maximise the likelihoods computed in the previous step.

E-step: find weights $\gamma_{t,k}$ encoding the probability of a point X_t^p to belong to a cluster k.

$$\gamma_{t,k} = \frac{\pi_k \prod_{p=1}^{P} \mathcal{N}(X_t^p | \mu_{p,k}, \Sigma_{p,k})}{\sum_{i=1}^{K} \pi_i \prod_{p=1}^{P} \mathcal{N}(X_m^p | \mu_{p,i}, \Sigma_{p,i})} \tag{2.3}$$

Where $p \in \{1 \dots P\}$ and P is the total number of parameters, $k \in \{1 \dots K\}$ and K is the total number of Gaussian clusters of means $\mu_{p,k}$, covariance matrix $\Sigma_{p,k}$ and mixing coefficient π_k.

M-step: for each cluster k, update its mean $\mu_{p,k}$, covariance matrix $\Sigma_{p,k}$ and mixing coefficient π_k.

$$\pi_k = \frac{\sum_{t=1}^{T} \gamma_{t,k}}{T} \tag{2.4}$$

$$\mu_{p,k} = \frac{\sum_{t=1}^{T} \gamma_{t,k} X_t^p}{\sum_{t=1}^{T} \gamma_{t,k}} \tag{2.5}$$

$$\Sigma_{p,k} = \frac{\sum_{t=1}^{T} \gamma_{t,k} (X_t^p - \mu_{p,k})(X_t^p - \mu_{p,k})^\top}{\sum_{t=1}^{T} \gamma_{t,k}} \tag{2.6}$$

2.5.2. Reproducing

The learnt model is then used to reproduce the path given new positions of task parameters, using a process known as task parametrised Gaussian Mixture Regression (TP-GMR). Firstly, the GMM from each parameter is transferred back from the local coordinate frame of parameter p to the global frame of reference. Then, a weighted multiplication is performed between the Gaussians of each parameter to obtain the final Gaussian $\mathcal{N}(\mu_k, \Sigma_k)$:

$$\mathcal{N}(\mu_k, \Sigma_k) \propto \prod_{p=1}^{P} \mathcal{N}\left(A_p \mu_{\boldsymbol{p},k} + \boldsymbol{b_p}, A_p \Sigma_{\boldsymbol{p},k} A_p^\top\right) \tag{2.7}$$

Once the joint probabilities, i.e. K Gaussians components, are obtained, regression is performed to obtain a Gaussian for each time step. To perform regression, the demonstration point dimensions were decomposed into an input and outputs. The input dimension $\mathcal{I}$ corresponded to the time step t dimension. The output dimensions $\mathcal{O}$ describe the path, such as end-effector position or velocity. The Gaussian components are decomposed into

$$\mu_k = \begin{bmatrix} \mu_k^{\mathcal{I}} \\ \mu_k^{\mathcal{O}} \end{bmatrix}, \Sigma_k = \begin{bmatrix} \Sigma_k^{\mathcal{I}} & \Sigma_k^{\mathcal{I}\mathcal{O}} \\ \Sigma_k^{\mathcal{O}\mathcal{I}} & \Sigma_k^{\mathcal{O}} \end{bmatrix} \tag{2.8}$$

Then, the conditional probability $P(\xi_t^{\mathcal{O}} \mid \xi_t^{\mathcal{I}})$ is calculated as follows:

$$P\left(\xi_t^{\mathcal{O}}|\xi_t^{\mathcal{I}}\right) \sim \sum_{k=1}^{K} h_k\left(\xi_t^{\mathcal{I}}\right) \mathcal{N}\left(\hat{\mu}_k^{\mathcal{O}}\left(\xi_t^{\mathcal{I}}\right), \hat{\Sigma}_k{}^{\mathcal{O}}\right) \tag{2.9}$$

where

$$\hat{\mu}_k^{\mathcal{O}}\left(\xi_t^{\mathcal{I}}\right) = \mu_k{}^{\mathcal{O}} + \Sigma_k{}^{\mathcal{O}\mathcal{I}}\Sigma_k{}^{\mathcal{I}-1}\left(\xi_t^{\mathcal{I}} - \mu_k{}^{\mathcal{I}}\right) \tag{2.10}$$

$$\hat{\Sigma}_k{}^{\mathcal{O}} = \Sigma_k{}^{\mathcal{O}} - \Sigma_k{}^{\mathcal{O}\mathcal{I}}\Sigma_k{}^{\mathcal{I}-1}\Sigma_k{}^{\mathcal{I}\mathcal{O}} \tag{2.11}$$

$$h_k\left(\xi_t^{\mathcal{I}}\right) = \frac{\pi_k \mathcal{N}\left(\xi_t^{\mathcal{I}}|\mu_k{}^{\mathcal{I}}, \Sigma_k{}^{\mathcal{I}}\right)}{\sum_{k=1}^{K} \pi_k \mathcal{N}\left(\xi_t^{\mathcal{I}}|\mu_k{}^{\mathcal{I}}, \Sigma_k{}^{\mathcal{I}}\right)} \tag{2.12}$$

The conditional probability constitutes of $\mu_t^{\mathcal{O}}$and $\Sigma_t{}^{\mathcal{O}}$where the former is considered to be the reproduced path point. The TP-GMM and TP-GMR form the basis of the contributions presented in this thesis.

2.6. Conclusion

In summary, this chapter describes the various range of programming technologies researched to program cobot's intuitively for flexible tasks. The cobot programming technologies are categorised into three features, that are communication, optimisation and learning features, and relevant research works on the defined features are reviewed in detail. Communication features enable a collaborative human operator to transfer intent or commands directly to a cobot, thus affecting its course of action to support collaboration. Optimisation features are algorithms developed by programmers on-line enabling a cobot to observe its collaborative operator and behave adaptively according to a pre-modelled optimised policy. Learning features allow a cobot to learn its own policy after receiving guidance from its collaborative operator. Learning from demonstration was predominantly of interest due to its combined potential of intuitive programming and flexible behaviour. In particular, more details were presented on task parametrised Gaussian Mixture Models as this project's core algorithm. TP-GMM's mathematical model was described in detail to further justify the research gaps and provide mathematical basis for chapter 3 and chapter 4.

Chapter 3: Visual Features as Task Parameters

3.1. Introduction

In this chapter, the first contribution in this thesis is presented. It includes a generic solution to the problem of identifying and detecting spatial task parameters for the task parametrized learning from demonstration algorithm. The developed solution filtered through generic visual features to identify ones that produce the closest path to the demonstration path. A simulation cobot arm was programmed to perform various industrial tasks using the generic developed algorithm.

3.1.1. Background

Task parameterized learning from demonstration (TP-LfD) algorithms model robot behaviour with respect to multiple task parameters. Instead of learning a single consistent behaviour from the demonstrations as in LfD, TP-LfD learns a behaviour dependant on environmental factors.

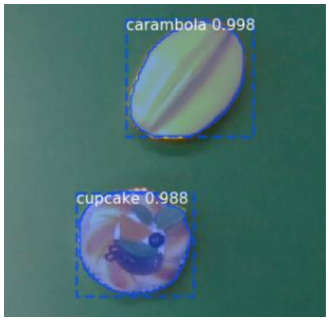
In most cases, task parameters are the positions of objects/items in space, also called frames of reference or frames. These objects are relevant to the task, such as a tool the robot uses, a part the robot has to pick, etc. For example, in a simple pick-and-place task, the task parameters are the position and orientation (6D position) of the part to be picked and the 6D position of the container the part is dropped in. Assume a user records a few (4-6) demonstrations where s/he varies the 6D position of the part and the container and manually moves the robot such that part is picked and placed in the container. TP-LfD learns the task such that if any of the part or the container moves, the robot's path will adjust accordingly. When a human is collaborating with a cobot in a task, such as handing tools, the human's hand also becomes a relevant "object" to the task. Therefore, one can also consider human hands to be one of the task parameters.

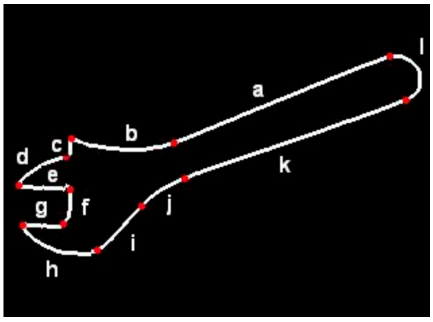
3.1.2. Task Parameter Detection

An object (and its 6D position) can be detected in a variety of ways. Classical techniques include motion capture and sticker markers. In motion capture, multiple prominent bulbs are placed on an object and easily detected and localised by a system of multiple cameras, such as in (Vogt et al., 2017). In stickers markers, a sticker with a prominent pattern is pasted on to the part and is detected and localised reliably by a 2D camera, such as in (Paxton et al., 2017; Perez-D'Arpino & Shah, 2017). However, these techniques are intrusive as they rely on extra hardware being placed on the objects of interest. Other non-intrusive object detection techniques involve image processing algorithms such as colour segmentation (Duque et al., 2019), contour matching (Rogowski & Skrobek, 2020) and cloud point matching (Y. Gu et al., 2018). However, the complexity of the shape and colour of industrial parts might render these algorithms non-reliable and difficult to implement. Some methods are designed based on machine learning techniques, such as support vector machine (SVM)-based classifiers, such as in (Dīnēshchandra Jōshī et al., 2020; Penumuru et al., 2020), and deep learning networks (Jia et al., 2020;

Park et al., 2020). These methods are reliable in detecting and localising objects in industrial scenes. However, they rely on a large amount of training data to function accurately, which introduces expensive processes of data collection and algorithm retuning for each new task. Figure 3.1 shows some examples of the object detection techniques previously mentioned.

This item has been removed due to 3rd Party Copyright. The unabridged version of the thesis can be found in the Lanchester Library, Coventry University.





(a) (b) (c)

Figure 3.1 Examples of object detection techniques: (a) using sticker markers (Perez-D'Arpino & Shah, 2017)*, (b) using Mask R-CNN* (Jia et al., 2020)*, (c) using contour matching* (Rogowski & Skrobek, 2020).

In this chapter, we present an algorithm that detects and localises task parameters while fulfilling industrial requirements: non-intrusive, reliable and generic.

3.1.3. Task Parameter Optimisation

Task parameters are usually selected manually. This could lead to sub-optimality, namely if the user choses a task-irrelevant frame or choses two frames that are redundant. Irrelevant frames are frames that are randomly occurring and of no relevancy to the task. Redundant frames are defined as a set of frames belonging to the same rigid object, i.e. with fixed relative positions with each other. Due to their fixed relative position, redundant frames will have the same GMMs after training. Accounting for all redundant and irrelevant frames in TP-LfD degrades algorithm performance as the path becomes falsely biased. Ideally, there should be one frame per task-relevant object.

To group redundant and eliminate irrelevant frames, Alizadeh et al. defined an importance score $F_{t,p}$ for a frame p at time step (point along the demonstration path) t (Alizadeh & Karimi, 2018; Alizadeh & Malekzadeh, 2017),

$$F_{t,p} = \frac{|\Sigma_{p,t}^{-1}|}{\sum_{i=1}^{P} |\Sigma_{i,t}^{-1}|}, \tag{3.1}$$

where P is the total number of frames and $\Sigma_{p,t}$ *is* the covariance matrix of the Gaussian of frame p at time step t. The importance score is the determinant of the inverse of the covariance matrix of a frame p at time step t divided by the sum of determinants over all frames. That means for frames with high values in the covariance matrix, (high variability) the determinant of the inverse will be low, and hence the importance score will be low. The Gaussians are obtained by training a TP-GMM to model the set of demonstration paths. Frames with equal importance score were deemed redundant and only one of

them was used for path reproduction (Alizadeh & Karimi, 2018). However, in a real-life situation with slight frame position disturbances, even redundant frames will not have exactly equal importance scores. Therefore, an error threshold should be introduced to account for slight frame mis-localisations. Moreover, Alizadeh and Karimi's algorithm necessitates the training of TP-GMM to obtain the covariance matrices before grouping redundant frames. However, with a large number of frames (more than ~50), the TP-GMM would fail to converge. Therefore, it is important that another algorithm is devised that groups redundant frames before training the TP-GMM.

Alizadeh and Malekzadeh used the same importance score to eliminate irrelevant frames (Alizadeh & Malekzadeh, 2017). Frames with an importance score below a certain threshold are considered irrelevant and eliminated. However, choosing an appropriate threshold is tricky in the cases when some frames have subtle yet important effects on the path. Moreover, this method does not eliminate redundant frames in case some are missed by the algorithm presented in (Alizadeh & Karimi, 2018). Therefore, it is important that another algorithm is devised such that it relies on closed loop feedback to choose the relevant frames while optimising TP-GMM performance.

Huang et al. used reinforcement learning to shift the position of frames until a task-specific cost function is minimised (Y. Huang et al., 2019). Moreover, they develop an automatic frame selection algorithm in which they identify which frames contribute most to minimising the cost function. They eliminate the frames that have a low influence on the learning which speeds up computation and improves performance. However, their approach does not tackle how to visually detect frames of reference, but rather they are specified as fixed positions with respect to the cobot's end-effector. This eliminates cases in which the frames are intrinsically defined on non-static objects.

The algorithm presented in this chapter achieves both goals: detecting and optimising task parameters. The algorithm is integrated in an end-to-end learning from demonstration system, called GenLfD, and validated in multiple industrial case studies.

3.2. Procedure

The GenLfD approach consists of two main procedures: training to generate a TP-GMM modelling the demonstration paths with respect to the frames of reference, and path reproduction based on TP-GMR when presented with frames of reference in new positions. For the training process, there are several critical steps: 1) capturing demonstration images, 2) recording demonstration paths, 3) detecting frames of reference from the images of the recorded demonstrations, 4) grouping redundant frames, 5) generating TP-GMM, and 6) eliminating irrelevant frames. In the reproduction process, the main steps are: 1) recording a new setting, 2) detecting frames from the new image of the setting, 3) matching them with relevant frames from the demonstrations, and 4) regenerating a path using TP-GMR. Figure 3.2 visualises the steps of the GenLfD algorithm.

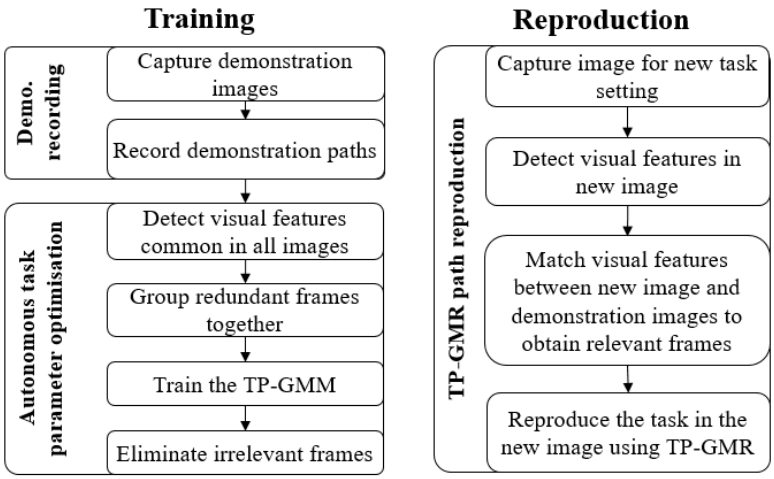


Figure 3.2 Overview of the GenLfD algorithm

3.2.1. Recording Demonstrations

Each demonstration entails the image of a scene and a path that the cobot needs to perform the task. In this chapter, the images of the demonstrations are recorded in 2-D to simplify the computation complexity of the approach, i.e., the top view of objects located on a surface. To record the path, the user saves a series of waypoints and the path is interpolated in between them. The default total number of path points interpolated is 200. This is similar to kinaesthetic teaching.

The path recorded is 5 dimensional: time step t, x-y-z coordinates and gripper state g. The x-y-z coordinates are measured with respect to a global frame of reference that can be randomly chosen. Gripper state g can be either 0 for open gripper or 1 for closed gripper. It is assumed that the cobot moves at constant speed. Therefore, the time step dimension t is set to be uniformly increasing throughout the path from 0.01 to 2.

Figure 3.3 shows an example of a recorded demonstration for a pick-and-place task. Four waypoints are recorded by the user: 1) a pre-grasping point right about the cube to be grasped with an open gripper, 2) a grasping point where the object is in the gripper with a closed gripper, 3) an intermediate point between the pick and place locations with a closed gripper, and 4) a placing point above the container with an open gripper. Table 3.1 shows the data recorded for these 4 way points. The path is interpolated as a straight line between the way points. Figure 3.4 shows a 2D projection of the demonstration path on the demonstration image.

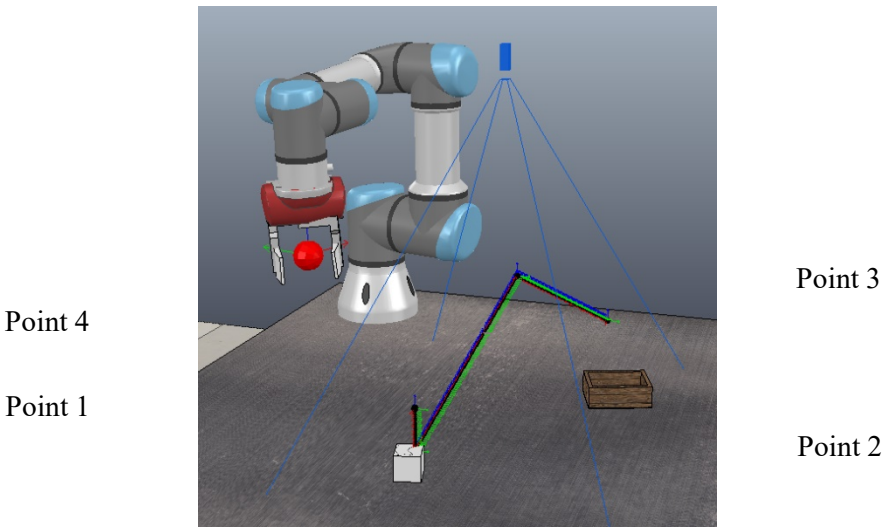


Figure 3.3 An example of recorded way points for a pick-and-place task as well as the interpolated path between the points.

Table 3.1 An example of the recorded data for the way point obtained from the simulation scene.

Dimension / Point	*t*	*x*	*y*	*z*	*g*
Point 1	0.01	0.0259	2.5444	0.725	0
Point 2	0.21	0.0243	2.5439	0.6750	1
Point 3	1.15	-0.0897	2.6257	0.85	1
Point 4	2.00	-0.255	2.6999	0.75	0

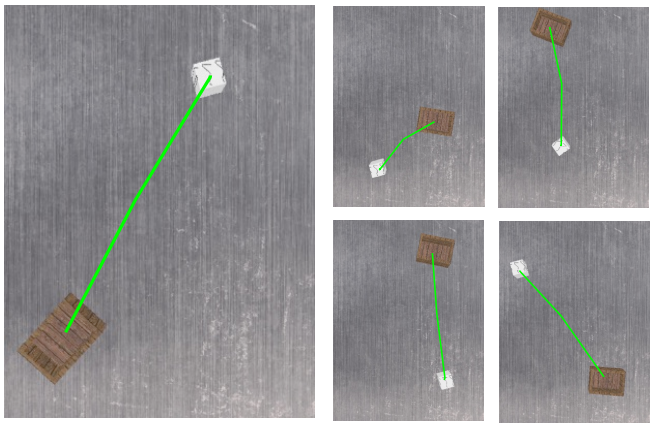


Figure 3.4 The 2D projection of the demonstration paths on the demonstration images for all 5 demonstrations.

In this algorithm, the number of demonstrations recorded is set to a default value of 5. The user is required to vary the positions of objects across demonstrations. Variations should cover as wide a range as possibly allowed in the task environment. This should be an educated decision done by the user who knows the task or object constraints. This variety helps the cobot learn a generalised model robust to changes in position of objects. Figure 3.4 shows examples of 5 recorded demonstration images and paths with varied object positions.

3.2.2. Detecting Visual Features

In an effort to simplify the process of frame detection for GenLfD, visual features are used to identify the frames of reference (task parameters). This presents a generic solution that can work for a wide range of objects without tweaking. Moreover, only a minimal setup of a 2D camera is required. Two types of visual features are detected: Speeded Up Robust Features (SURF) and hand features.

SURF features: Interest points are detected based on geometric information, such as corners, T-junctions and blobs. Such interest points are widely available in objects, which makes the GenLfD applicable with a wide range of object shapes and textures.

Descriptor vectors are calculated describing each interest point. The vectors are scale and rotation invariant which allows them to be matched robustly from different images (Bay et al., 2006). SURF features can be matched across different demonstration images even when objects vary orientation on the table.

Moreover, SURF features provide a balance between detection time and number of features retrieved (Tareen & Saleem, 2018) compared to other feature detectors. Therefore, enough features can be retrieved from task-relevant objects that serve as task parameters for the TP-GMM training. Figure 3.5 shows the detected SURF features from the demonstration images.

However, using SURF features presents a few limitations. Firstly, interest points are detected from the intensity values of pixels rather than the RGB value. That means, the feature detector does not take into account colour information. Therefore, identical objects of different colours cannot be differentiated using SURF features. Moreover, SURF features are not robust to reflective surfaces or shadows. This might require speciality industrial anti-shadow lighting to be used, depending on the texture of the objects involved.

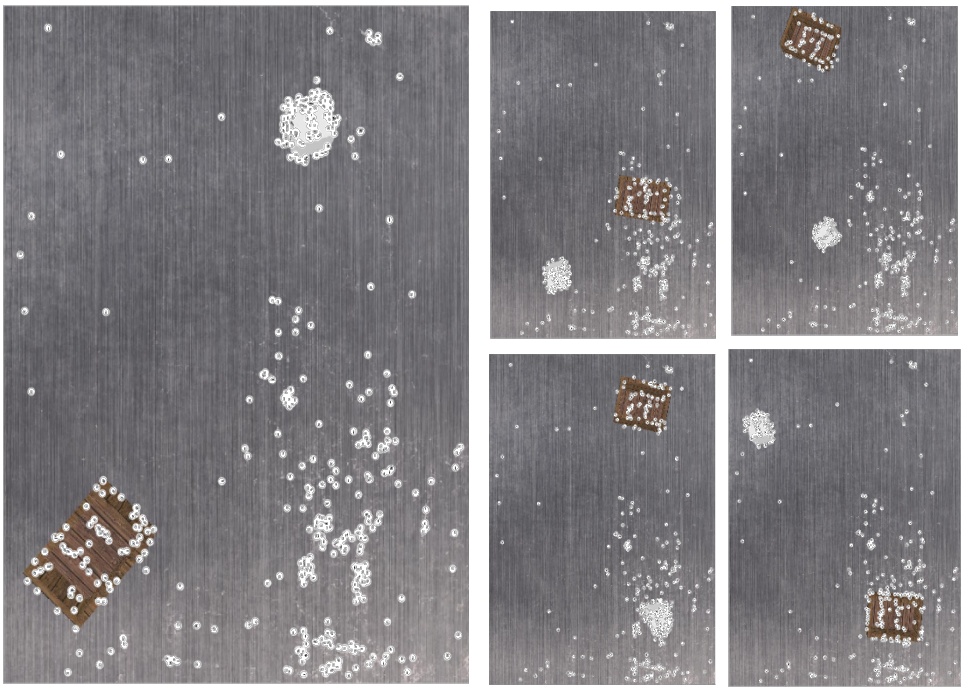


Figure 3.5 Detected SURF features from the demonstration images of the pick-and-place task.

Matching SURF features is done using exhaustive nearest neighbour search method (Muja & Lowe, 2009). Features that are matched across all the images of the demonstrations are kept. If a feature is not found in at least one of the images, it is eliminated since TP-GMM can only be trained if a feature is found in all demonstration images. Figure 3.6 shows the matched features between the demonstration images of the pick-and-place task. The total number of matched features is defined as *P*.

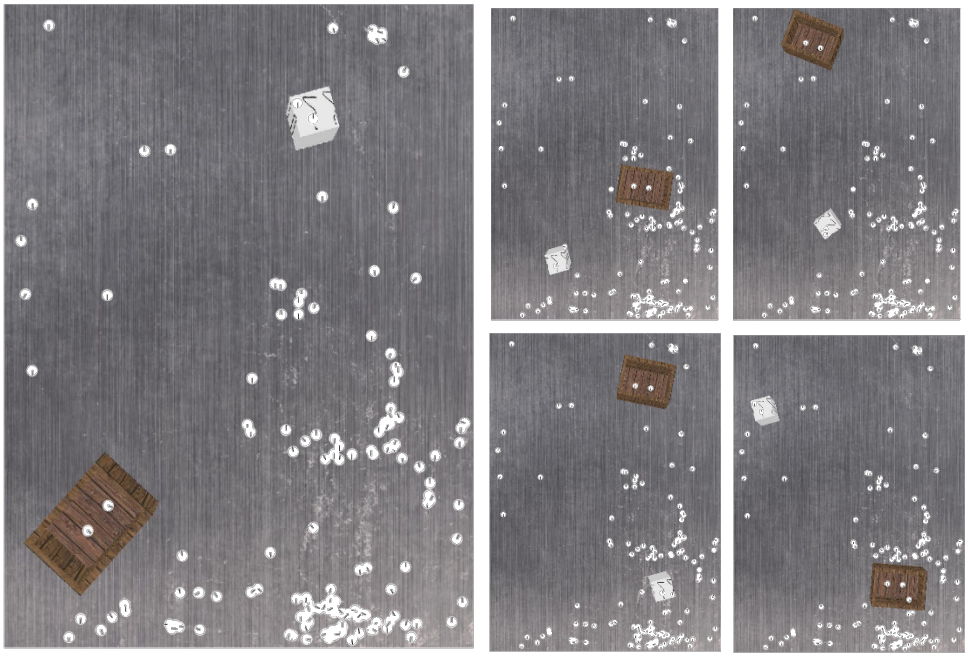


Figure 3.6 Matched SURF features from the demonstration images of the pick-and-place task.

Once a SURF feature is matched in all demonstration images, it is saved as a task parameter to be used in training TP-GMM. Each task parameter p is characterised by a pixel position vector $b_{p,m}=\{x,y\}_{p,m}$ and an orientation $\alpha_{p,m}$, where $m \in \mathbb{N}\{1,...,M\}$, the total number of the demonstration images and $p \in \mathbb{N}\{1,...,P\}$ is the total number of task parameters (Figure 3.7). From the orientation, a 2x2 rotation matrix $A_{p,m}$ is calculated.

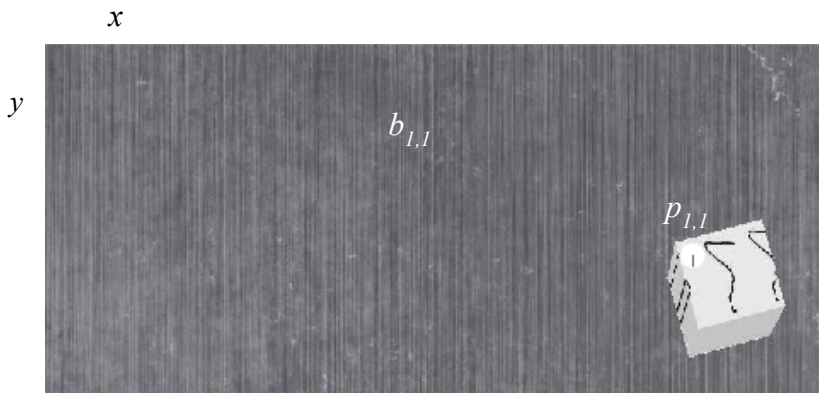


Figure 3.7 an example of task parameter 1 from demonstration 1 and its pixel position vector.

The units of the demonstration path's x-y-z coordinates are real-life measurement units, e.g. centimetres, metres, etc. Therefore, the task parameter position units are converted from pixels to real-life units based on the position of the camera with respect to the global reference frame. Moreover, the orientation matrix is also adjusted to be consistent with the real-life orientation convention.

To convert the *x-y-z* coordinates of frames of reference from pixel positions to real-life coordinates, the following steps are followed:

1. Camera calibration parameters are obtained:
   a. $\{x_{cam\_pos}, y_{cam\_pos}, z_{cam\_pos}\}$ be the coordinates of the camera in space with respect to a randomly chosen global reference frame, on the surface of the task table
   b. $\{resol_x, resol_y\}$ be the image pixel resolution in the direction of $\{x, y\}$ as defined in space
   c. α_{proj} be the perspective angle of the camera lens
2. The real length of the surface displayed in the image is calculated as such:

$$real_x = 2 \times z_{cam\_pos} \times \tan(\alpha_{proj}/2) \tag{3.2}$$

$$real_y = real_x \times \frac{resol_y}{resol_x} \tag{3.3}$$

3. Each frame's position $b_{p,m}=\{x,y\}_{p,m}$ is converted from pixel to real-life coordinates:

$$x_{p,m} = real_x \times \frac{\left(\frac{resol_x}{2} - x_{p,m}\right)}{resol_x} + x_{cam\_pos} \tag{3.4}$$

$$y_{p,m} = real_y \times \frac{\left(\frac{resol_y}{2} - y_{p,m}\right)}{resol_y} + y_{cam\_pos} \tag{3.5}$$

Hand features: hand features are detected using a pre-trained YOLOv3 neural network (Redmon & Farhadi, 2018). YOLOv3 is a real-time deep convolutional neural network that identifies a wide range of pre-trained objects in images/videos. YOLOv3 surveys the image as a whole before identifying classes in their bounding boxes while accounting for the full image context. A hand detection is a square bounding box on the 2D image. If multiple hands are detected in a demonstration image, there is no differentiating factor between them except a detection confidence score. Matching hand features across demonstration images is not possible. Therefore, the hand of the highest confidence score from each demonstration image is considered and the rest of the hand features are ignored. This is an acceptable assumption since it is assumed that the cobot is only collaborating with one other human operator. Moreover, if one of the operator hands is mistakenly detected instead of the other while accidentally in the image frame, the operator is encouraged to remove his hand from the camera's field of view. However, if both of the operator's hands are present purposefully, then it is assumed they are both operating on the same object and hence either of them is okay to detect. Alternatively, in the future, we can add an option in the programming GUI for the operator to choose whether the right-most hand or the left-most hand in the image is to be detected.

Moreover, the hand bounding box is not oriented. Therefore, the orientation of the hand is considered to be equal in all the images and is set to zero degrees. This is an acceptable assumption since it is likely that the human operator stands in a consistent direction with respect to the cobot. In the handover task

shown in Figure 3.8, the hand is detected and then combined with the SURF features to form the total set of task parameters.

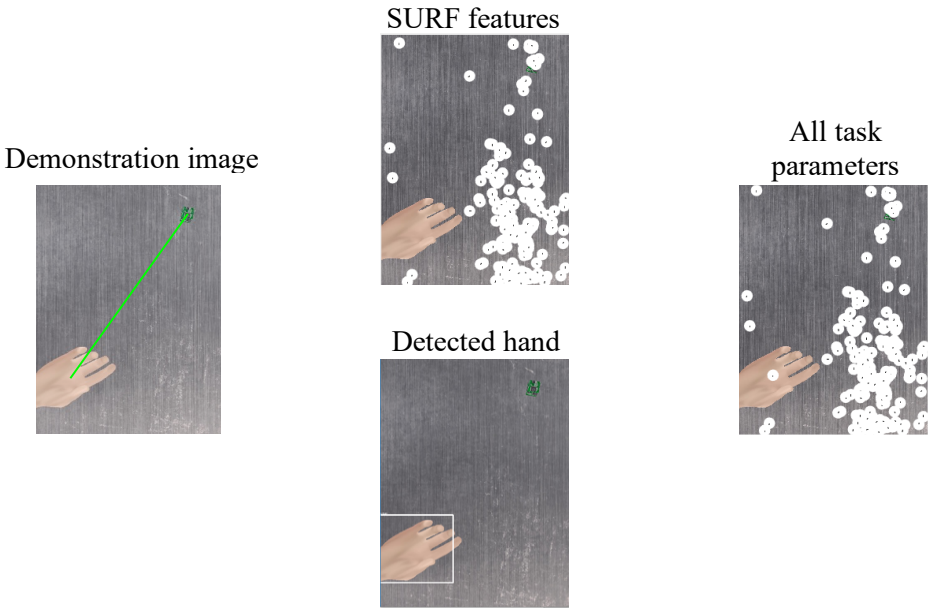


Figure 3.8 In a handover task, the robot is going to hand the circuit board on the table to its human partner. The hand features are detected separately using YOLOv3 and then combined with the SURF features to form the total set of task parameters.

3.2.3. Grouping Redundant Frames

Redundant frames are frames that belong to the same rigid object. Due to their fixed relative position with respect to each other, the obtained GMM for each frame are identical. When reproducing a path using the GMMs of all redundant frames, the path will be falsely biased towards the largest group of redundant frames. Therefore, redundant frames must be identified and grouped together as one *object* and only one frame should be used in the TP-GMM and TP-GMR algorithms. This one frame from the *object* is called the *lead frame* and is chosen based on the highest frame detection confidence value (a parameter returned by the SURF detector algorithm).

Redundant frames are identified based on their relative positions with respect to each other, following these steps:

1. The distance $d_{jp,m}$ and relative orientation $\alpha_{jp,m}$ between two frames j and p in a demonstration m are calculated.

$$d_{jp} = \{d_{jp,m} \; \forall \; m \in \mathbb{N}\{1, \dots, M\} \; such \; that \; d_{jp,m} = \sqrt{(x_{j,m} - x_{p,m})^2 + (y_{j,m} - y_{p,m})^2} \quad (3.6)$$

$$a_{jp} = \{a_{jp,m} \; \forall \; m \in \mathbb{N}\{1, \dots, M\} \; such \; that \; a_{jp,m} = \; (\alpha_{j,m} - \alpha_{p,m}) \quad (3.7)$$

2. The standard deviation $\sigma(d_{jp})$ and the mean $\mu(d_{jp})$ of the distance d between frames j and p across all demonstrations are calculated.

$$\mu(d_{jp}) = \frac{1}{M}\sum_{m=1}^{M} d_{jp,m} \tag{3.8}$$

$$\sigma(d_{jp}) = \sqrt{\sum_{m=1}^{M}\left(d_{jp,m} - \mu(d_{jp})\right)^2 \Big/ M} \tag{3.9}$$

3. The standard deviation $\sigma(\alpha_{jp})$ and the mean $\mu(\alpha_{jp})$ of the relative orientation α between frames j and k across all demonstrations are calculated.

$$\mu(\alpha_{jp}) = \frac{1}{M}\sum_{m=1}^{M} \alpha_{jp,m} \tag{3.10}$$

$$\sigma(\alpha_{jp}) = \sqrt{\sum_{m=1}^{M}\left(\alpha_{jp,m} - \mu(\alpha_{jp})\right)^2 \Big/ M} \tag{3.11}$$

4. If the standard deviation divided by the mean, $\sigma(d_{jk})/\mu(d_{jk})$ and $\sigma(\alpha_{jk})/\mu(\alpha_{jk})$ are both below a certain threshold ε (see Equation 4), the frames are deemed redundant.

$$if\ \frac{\sigma(d_{jk})}{\mu(d_{jk})} < \varepsilon \cap \frac{\sigma(a_{jk})}{\mu(a_{jk})} < \varepsilon \tag{3.12}$$

5. A P x P redundancy matrix *Redun* is defined to represent the redundancy relationship (which two frames are redundant) between frames, where P is the total number of frames. If frames j and k are redundant, $Redun_{jk}$ is set to 1.

$$then\ Redun_{jk} = Redun_{kj} = 1 \tag{3.13}$$

6. When the matrix is completed, frames that are redundant are grouped together into one *object*. From each *object*, the frame with the highest detection confidence value (a score returned by the SURF detection algorithm) is set to be the *lead frame*. The *lead frame* will be used in training the TP-GMM.
7. If the total number of obtained lead frames is more than the maximum number of *lead frames* allows, the threshold is increased by an increment and steps 4 to 7 are repeated again.
8. If the total number of obtained lead frames is less than the maximum number of lead frames allows, the process is complete.

The maximum number of *lead frames* allowed is set to be 25. This number is a generous estimation of the number of objects in a scene. The threshold ε is initialised as 0.05 and increased by increments of 0.01. Figure 3.9 shows that as the threshold ε is increased, the number of *lead frames* decreases.

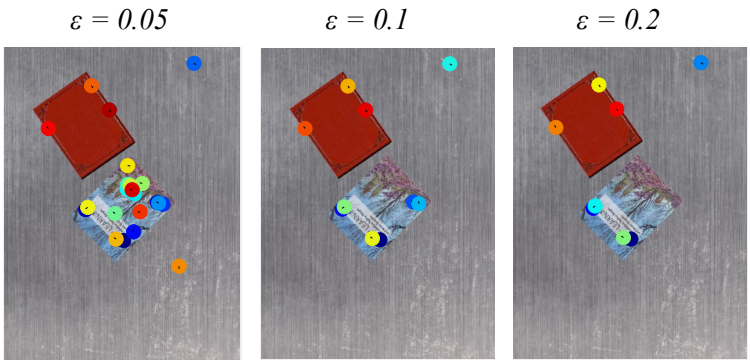


Figure 3.9 As threshold ε increases, the total number of lead frames obtained decreases.

Figure 3.10 shows the grouped redundant frames in the pick-and-place task. The frames of the same colour belong to the same *object*. There are multiple recognizable errors:

1. Error 1: The red frame is slightly rotated compared to its matches in the other demonstrations. Therefore, the algorithm failed to detect that it belongs to the same object, i.e. the container, as the yellow frame.
2. Error 2: The dark blue frame is mistakenly matched on a different corner of the cube in the second demonstration. Therefore, the algorithm failed to detect that it belongs to the same object, i.e. the cube, as the light blue frame.
3. Error 3: The orange frame belonging to the table failed to be grouped with the table object frames (in blue) since it is an outlier in the last demonstration, i.e. it was falsely detected on the container.

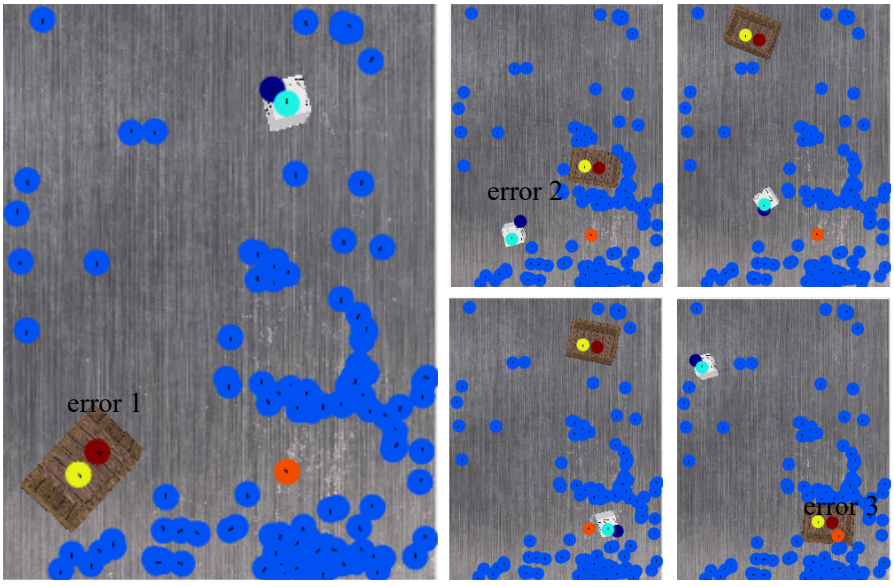


Figure 3.10 The grouped redundant frames in the pick-and-place task and the three potential types of errors encountered.

However, such errors do not affect the final result of the algorithm since any erroneous frames will be eliminated by the reinforcement learning algorithm.

3.2.4. TP-GMM Training

After grouping redundant frames, effectively reducing the total number of frames, the TP-GMM is trained. More details about TP-GMM can be found in chapter 2.

It is important to note that the frames of reference are 2-dimensional (*x, y*), whereas the path data is 5-dimensional (*t, x, y, z, g*). Therefore, when transforming the path data from the general frame of reference to each parameter's frame of reference, only the *x*-*y* coordinates are changed whereas the other dimensions remain untransformed. The time *t* and gripper state *g* dimensions do not vary with respect to each of the frames, since they are non-spatial dimensions. It is assumed that the z dimension also does not vary since the objects are placed on a table, so their height is constant across the demonstrations. This is a safe assumption since it is common for industrial tasks to involve table-based objects. However, it cases where objects do vary height and the *z* dimension is not fixed, then it needs to be detected using a depth sensor along with the *z* dimension of task parameters. From the TP-GMM training, we obtain a GMM modelling the path data with respect to each frame. Figure 3.11 shows an example of the demonstration paths with respect to p_1 which is associated with the cube in the pick-and-place task. Moreover, Figure 3.11 shows the GMM obtained modelling the path points with respect to p_1, which lies on the cube. As can be seen in Figure 3.11, the GMM constitutes of four Gaussian components, one of which is very small and close to p_1. This figure shows an example of the result of modelling path points with Gaussian distributions.

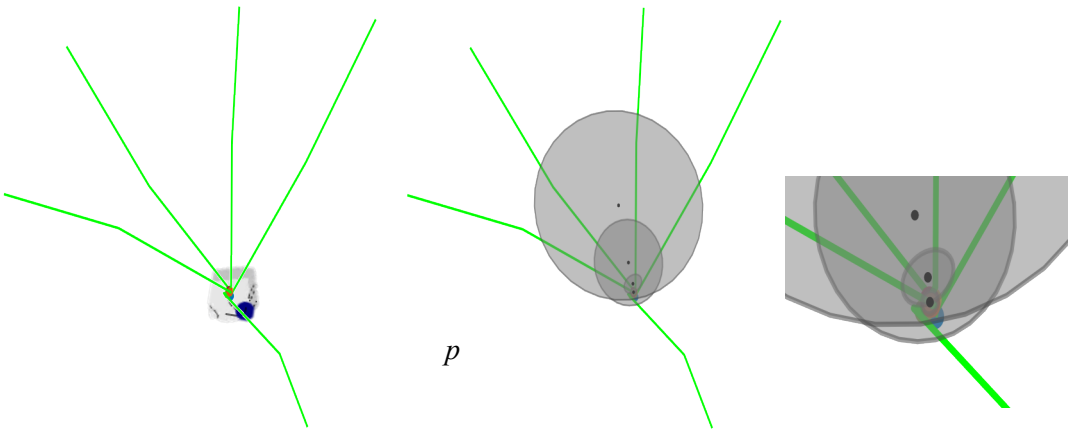


Figure 3.11 The paths with respect to p_1 as well as the GMM obtained modelling these paths.

These GMMs are used to reproduce the paths in new situations where the frames vary positions. However, some frames might be identified as orientation-less, which are frames whose orientation does not affect the functionality of the task paths. In that case, the path points are modelled as ring GMMs. Figure 3.12 shows the ring Gaussian modelling of the path points with respect to p_1, which is identified as an orientation-less frame.

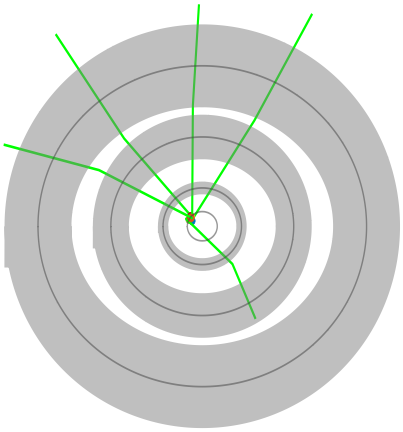


Figure 3.12 The ring GMM modelling for p_1, which is identified as an orientation-less frame.

3.2.5. Removing Irrelevant Frames

Irrelevant frames are frames that are not related to the path's function. For example, when detecting SURF features in an image, some of the features might be extracted from the background, from noise or from useless clutter. The performance of the TP-GMR is highly dependent on the set of frames given. If any of the frames are task-irrelevant, the performance of TP-GMR will deteriorate and the reproduced path might be unsatisfactory. Therefore, a reinforcement learning-based algorithm is designed to enhance the reproduced path by optimising the choice of frames used. Reinforcement learning enables finding the optimal frames, from a discrete set, without performing exhaustive search and while leveraging experience and previous results. Given the expected number of relevant frames, set to a default of $nbRelev = 2$, the algorithm identifies a set of $nbRelev$ frames that produce the best path reproduction. The reinforcement learning problem is formulated as follows:

State: The state of the environment is the reproduced path generated in each iteration of the algorithm

Action: The action set is the various frame combinations that can be chosen to reproduce the path

Model: The environmental model is the GMMs obtained from training the TP-GMM since they provide a mapping between the chosen frames and the reproduced path

Reward: The reward r is a function that assesses how close the reproduced path is to the demonstration path (ground truth)

Policy: The policy π is a function that describes the probability of a frame being used in the next path reproduction, based on the previous performance of each frame

The reinforcement learning algorithm is performed in the following steps:

1. Each lead frame p is given a relevancy probability, $probRelev_i$ that signifies how likely it is that a frame is relevant to the task, i.e. the probability that it should be used to reproduce the path. The goal of this algorithm is to maximise the values of $probRelev_i$ (policy) for relevant frames (action set) such that the generated path (state) is the most similar to the

demonstration path (maximum reward). Initially, the *probRelev*s are assigned equal values summing up to 1. Therefore, given *P* is the total number of lead frames,

$$probRelev_p = {}^{1}/_{P} \tag{3.14}$$

2. The algorithm is trained over *iterTot* number of iterations. The value of *iterTot* is set to a default value of *10PxnbRelev*. In each iteration *iter*, a disturbance $\Delta probRelev_{iter}$ to the relevancy probability is introduced as part of the exploration tactic in reinforcement learning. This results in temporary relevancy probability $\lambda probRelevs$ that are used as the policy π in iteration *iter*. The $\lambda probRelev$ for all lead frames sum up to 1. Given *rand* is a random number between 0 and 1,

$$\Delta probRelev_{iter} = \left(rand_p - probRelev\right) \times 0.05 \tag{3.15}$$

$$\lambda probRelev_{iter} = (probRelev + \Delta probRelev_{iter}) / \sum (probRelev + \Delta probRelev_{iter}) \tag{3.16}$$

3. In each iteration *iter*, a random demonstration *m* is chosen. The *nbRelev lead frames* to be used in path reproduction, i.e. action to be performed, is based on the values of $\lambda probRelev$, i.e. the policy. The *nbRelev* frames of maximum $\lambda probRelev$ are used to reproduce the path in demonstration *m.* Given the lead frames with maximum *probRelev,* the path, i.e. the state, is reproduced using the model, i.e. the GMMs or ring GMMs of these frames. For example, if the number of relevant frames is 2, then the 2 *lead frames* with the highest value of $\lambda probRelev$ at a certain iteration *iter* are used in TP-GMR.
4. Once a path, i.e. the state, is generated, a *cost* is calculated as the average of the Euclidean distance between the points of the reproduced path and the demonstration path for demonstration *m*. The cost measures the similarity between the reproduced path and the demonstration path, as a means of assessing the quality of the reproduced path and the policy at *iter*. Given that *T* is the total number of points in a path, (x_t, y_t) and $(X_{t,m}, Y_{t,m})$ are the *x*-*y* coordinates of the path point at time step *t* on the reproduced path and demonstration *m*'s path, respectively,

$$cost_{iter} = \frac{1}{T} \sum_{t=1}^{T} \sqrt{(x_t - X_{t,m})^2 + (y_t - Y_{t,m})^2} \tag{3.17}$$

5. For every *iterPeriod* number of iterations, the costs are normalised between 0 and 1. The value of *iterPeriod* is set to a default of *P*. Then, the reward *r* is calculated using the normalised costs, such that,

$$r_{iter} = sigmoid(-5 \times cost_{iter}) \tag{3.18}$$

6. For every *iterPeriod* number of iterations, the value of *probRelev* is updated based on the rewards obtained from the different disturbances $\Delta probRelev$. Based on the exploitation tactic in reinforcement learning, the policy *probRelev* is shifted towards the values of

$\lambda probRelev$ that yielded higher reward values in the last *iterPeriod* set of iterations. The value of *probRelev* slowly converges to the *lead frames* obtaining to highest reward value, i.e. the relevant *lead frames*. Figure 3.13 shows the decreasing value of the normalised cost as well as the increasing value of the reward obtained while optimising the pick-and-place tasks.

$$probRelev = probRelev + \sum_{iter} r_{iter} \times \Delta probRelev_{iter} \quad (3.19)$$

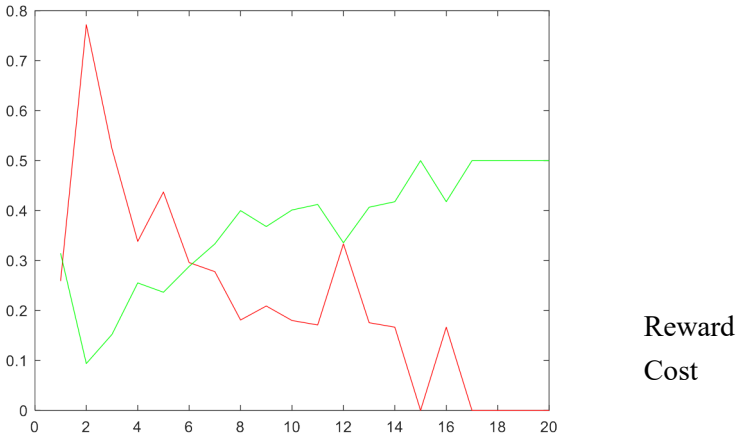


Figure 3.13 The value of the reward and normalised cost as a function of the number of iteration periods for training the pick-and-place task.

It is expected that the path reproduced using the relevant frames identified will be more similar to the demonstration path than the path reproduced using all lead frames. Figure 3.14 shows the difference between the demonstration path (green), the paths reproduced using the relevant frames identified (white) and that using all lead frames (red).

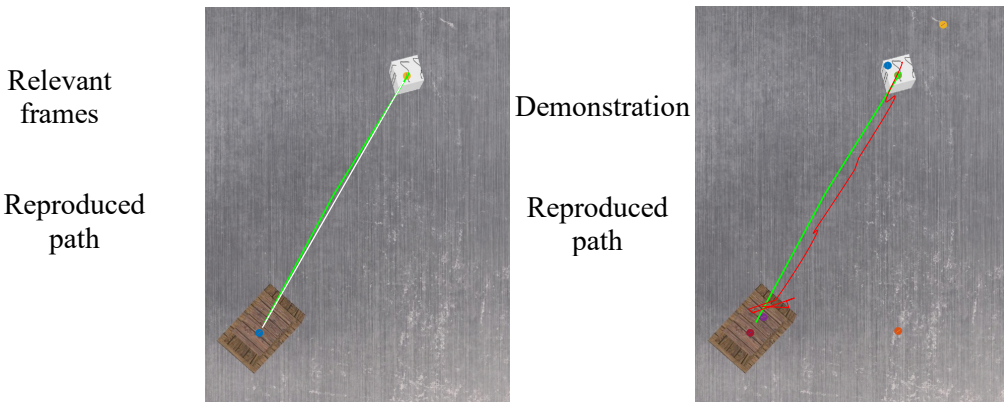


Figure 3.14 The demonstration path (green) and the paths reproduced using the relevant frames identified (white) and that using all lead frames (red).

3.2.6. Path Reproduction

After determining which frames are relevant amongst the lead frames, the cobot can apply TP-GMR to reproduce the path in a new scenario. The reproduction process is as follows:

1. An image is captured in the new setting, i.e. similar images to those in the demonstrations but with objects varying positions.
2. Visual features (SURF and hand features) are extracted from the new image. Figure 3.15 shows the SURF features detected from the new image.
3. The visual features in the new image are matched with the relevant frames from the demonstration images. This is done to identify the new locations in which the relevant frames are in the new image. Figure 3.15 shows the relevant frames successfully detected in the new image.
4. The GMMs or ring GMMs of the relevant frames are used to reproduce the path in the new image, using TP-GMR. More technical details on TP-GMR are found in chapter 2. Since the frames in this pick-and-place task are orientation-less, an adjusted TP-GMR algorithm is performed. Figure 3.15 shows the reproduced path in the new image successfully accomplishing the pick-and-place task.

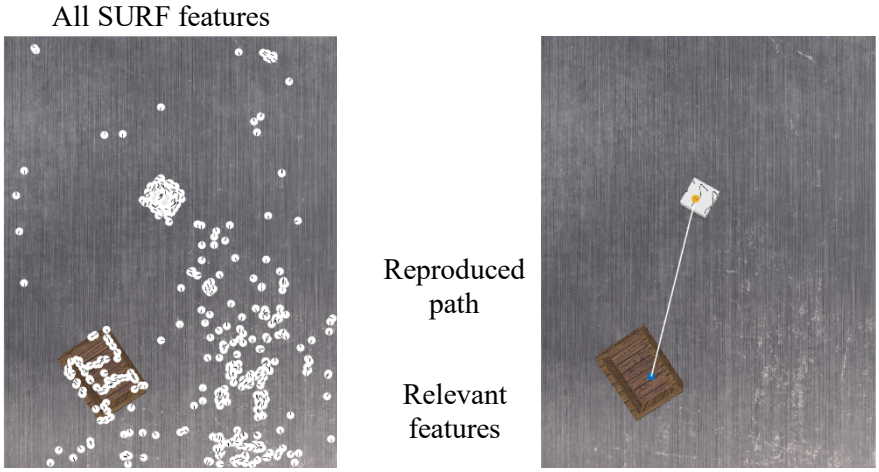


Figure 3.15 A visualisation of the steps performed to reproduce the task path in an image of a new setting.

3.3. Results

3.3.1. Varying number of lead frames

In this subsection, the performance of the reinforcement learning algorithm is investigated as the number of lead frames are varied. The purpose of this investigation is to test the robustness of the algorithm against a higher number of irrelevant frames. The experiment was performed on the synthetic data provided by Calinon (Calinon, 2016). Figure 3.16 show the four demonstrations provided of a path going from one frame to another as well as the path reproduction. One of the frames varies position across demonstrations.

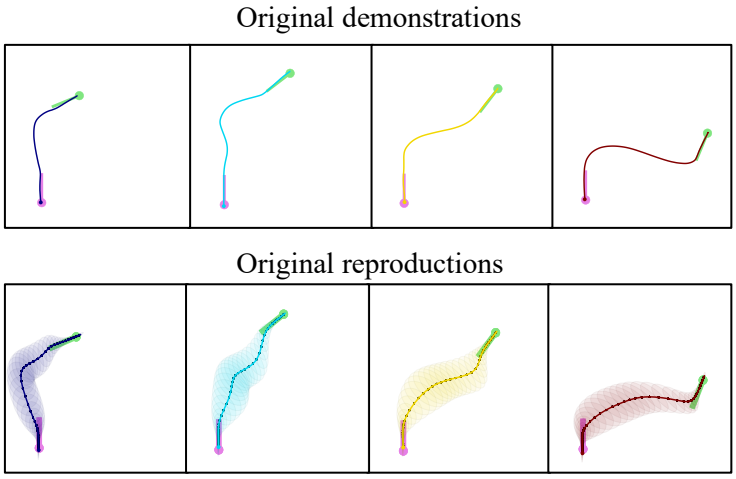


Figure 3.16 Demonstration data provided by Calinon (Calinon, 2016).

Irrelevant frames where added randomly to the original demonstrations such that the total number of *lead frames* was varied between 5 and 35 in increments of 5. Figure 3.17 shows the demonstrations with the concatenated 33 extra frames as well as the path reproductions.

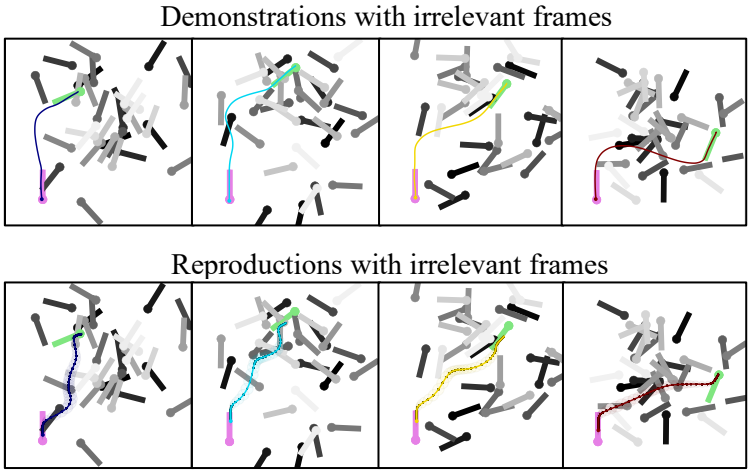


Figure 3.17 Demonstration data and reproduction paths with the additional 33 irrelevant frames.

Table 3.2 Success rate of reinforcement learning algorithm as a function of the number of lead frames.

Number of lead frames	Success rate (%)
5	85
10	90
15	100
20	100
25	90
30	95
35	100

The reinforcement learning algorithm was run 20 times on each number of *lead frames*. The reinforcement learning algorithm had to identify the 2 relevant frames. Table 3.2 shows the success rate for each number of *lead frames*.

The results show that the algorithm is capable of handling a large numbers of lead frames, i.e. scenes of high complexity with various clutter. The algorithm is robust and its performance is independent of the number of lead frames. That is mainly due to the fact that the parameters of the reinforcement learning algorithm are parametrized with respect to the number of lead frames. Therefore, when the number of lead frames varies, the parameters adjust in order to maintain a good performance of the algorithm. For example, the number of iterations in every period is equal to the number of lead frames and the total number of iterations is equal to the number of lead frames multiplied by the number of relevant frames multiplied by 10.

3.3.2. Simulation Results

To assess the results of the algorithm in chapter 3, industrial tasks were designed such that the relevant frames are oriented. This is done by maintaining a consistent orientation of relevant objects with a portion of the task path. The simulation scenes were built in CoppeliaSim EDU. The following tasks were tested:

Task 1 - Sorting: Given two circuit boards and two containers, the robot task is to pick the smaller circuit board and place it in the green container. The containers are in a fixed positions where as the circuit boards vary position, but not orientation. Task 1 shows that GenLfD can be used to program conditional industrial operations such as sorting. If two pick-and-place tasks are learnt, GenLfD is capable of identifying which task corresponds to which object by identifying the visual features of that object in an image.

Task 2 - Handover: The cobot needs to pick up an object, in this case a circuit board, and hand it to a human hand. Both the hand and the circuit board vary positions. Task 2 shows that GenLfD can program tasks involving human operators such as simultaneous or supportive tasks, defined in chapter 1.

Task 3 - Glue book spine: Given a book on a table, the cobot is required to apply glue to the book spine in a straight line. This is a task that could aid an operator before joining a book with a hard cover. The book and its associated hard cover vary positions on the table.

Task 4 - Pick-and-place: The robot is required to pick an object, a cube, and place it in a box. Both the cube and the box vary positions on a table. This could also be analogous to peg-in-hole or assembly tasks.

Six demonstrations are recorded for each task where the objects are in variable positions. Five of these demonstrations are used for training, and one of them is used for validation. The following parameters were set as defaults during training and were not changed between tasks:

- Number of demonstrations $M = 5$
- Number of Gaussian components $K = 4$
- Number of relevant frames *nbRelev* = 2

Figure 3.18 shows the results on the four different tasks in the various stages of the algorithm. The first row shows the grouped redundant frames in the validation image of each task. The frames of the same colour belong to the same *object*. It can be observed that even though some frames belong to the same *object*, they sometimes failed to be identified as redundant, such as in the circuit boards in Task 1, 2 and 4 and the book in Task 3. That is due to the book's complex design and some erroneous frame mislocalisation. This does not pose a problem in the results since the reinforcement learning algorithm will filter through the frames and choose the ones with the least mislocalisation error.

The second row in Figure 3.18 shows the result of training the TP-GMM and reproducing the path (in red) using all the *lead frames*. It can be observed that the reproduced path did not always achieve the goal of the demonstration path (in green). For example, in Task 1, the reproduced path failed to start from the centre of the circuit board. In Task 2, the reproduced path failed to reach the hand of the user. In Task 3, the reproduced path did not trace the book's spine but was rather offset. In Task 4, the reproduced path failed to reach the box. This error is due to the use of irrelevant and redundant frames in the TP-GMR. However, this error does not pose a problem in the final results since the reinforcement learning algorithm will eliminate the irrelevant frames.

The third row in Figure 3.18 shows the results of the reinforcement learning algorithm. In all cases, the relevant frames were successfully identified. An improvement in the reproduced path using the relevant frames only (in white) is noticed compared to the reproduced path using all lead frames (in red). For example, in Task 1, the white path successfully starts from the circuit board. In Task 2, the white path is slightly closer to the hand even though it still did not fully reach the final path destination. In Task 3, the white path successfully traces the book's spine very similar to the demonstration path (in green). In Task 4, the white path successfully reaches the box.

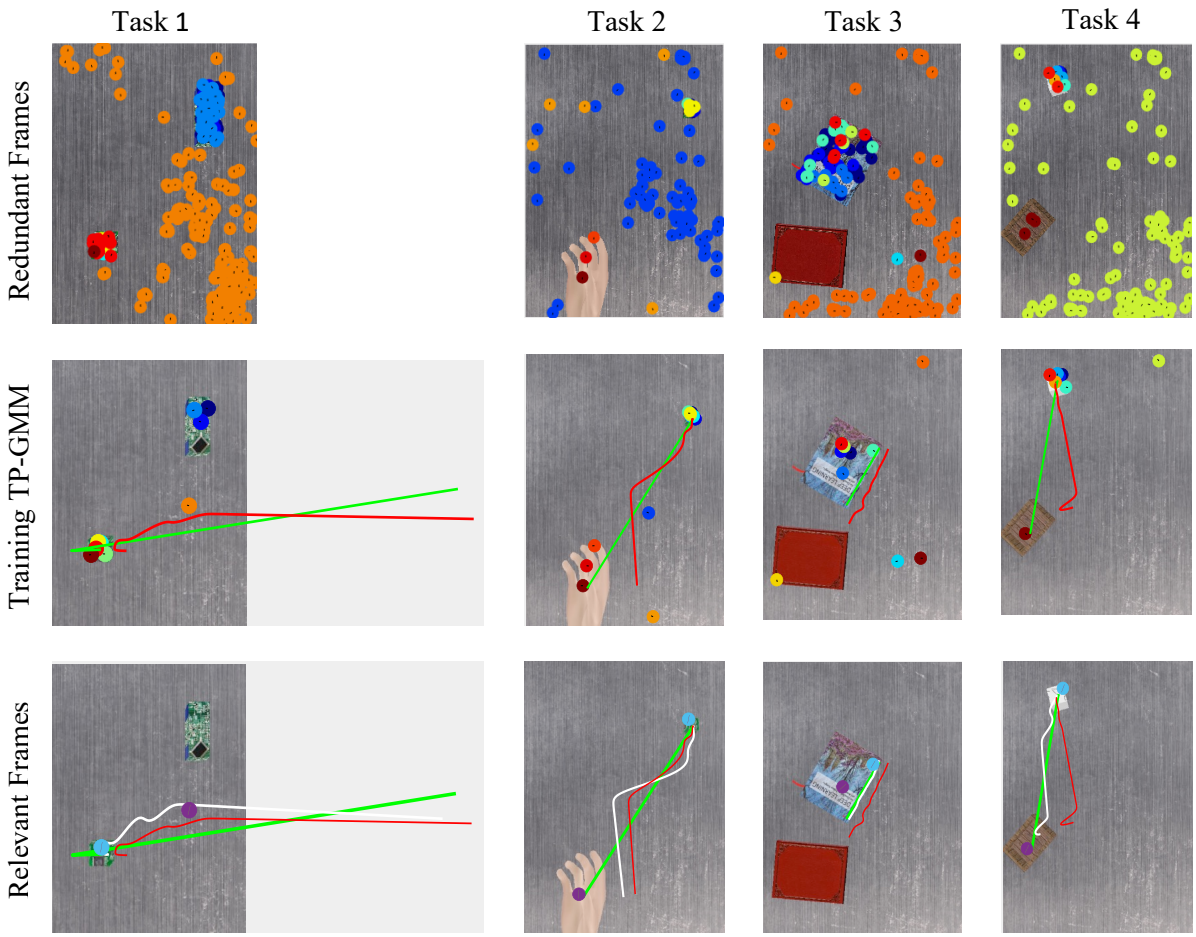


Figure 3.18 The results on the four tasks, at the different stages of the algorithm.

Table 3.3 Distances between the reproduced paths and the demonstration path.

Task	d_{total}^{all}	d_{total}^{rel}	d_{start}^{all}	d_{start}^{rel}	d_{end}^{all}	d_{end}^{rel}
Task 1	0.0381	0.0235	0.0208	0.0062	0.0268	0.0286
Task 2	0.0250	0.0236	0.0049	0.0043	0.0838	0.0688
Task 3	0.0238	0.0016	0.302	0.0017	0.0147	0.0020
Task 4	0.0344	0.0124	0.0064	0.0035	0.0846	0.0285

Table 3.3 shows the distances between the reproduced paths and the demonstration path for all four tasks. d_{total}^{all} is the average distance between all the points on the reproduced path using all lead frames and the demonstration path. d_{total}^{rel} is the distance between all the points on the reproduced path using relevant frames and the demonstration path. d_{start}^{all} is the distance between the first point on the reproduced path using all lead frames and the demonstration path. d_{start}^{rel} is the distance the first point on the reproduced path using relevant frames and the demonstration path. d_{end}^{all} is the distance between the last point on the reproduced path using all lead frames and the demonstration path. d_{end}^{rel} is the distance the last point on the reproduced path using relevant frames and the demonstration

path. The results show that overall the reproduced path using relevant frames is closer, i.e. has less value of *d* than the reproduced paths using all lead frames.

Moreover, Task 1, 2, 3 and 4 were run a 100 times each with the objects varying in positions randomly. The reproduced paths were surveyed manually and two different errors were identified. The percentage of occurrence of each type of error is shown in Table 3.3.

- Error 1 - Mis-locating a frame; when a frame is falsely detected in a wrong location. This error could be common in object with symmetries or repetitive patterns.
- Error 2 - Unsatisfactory TP-GMR; when the path reproduced using TP-GMR does not fulfil the task requirements e.g. the destination is not reached. This is due to the limitations of the TP-GMR and could be improved using El Zaatari et al. (El Zaatari et al., 2021) or the works of Sena et al. (Sena et al., 2019).

Table 3.4 Percentage of occurrence of the different error types in each task.

	Task 1	Task 2	Task 3	Task 4
Error 1	0	6%	3%	0
Error 2	50%	12%	16%	40%

Since the hand in Task 2 is synthetic (simulation, not real), that decreased the detection rate of the YOLOv3 algorithm cause Error 1 to appear as due to the high complexity of the object, i.e. the book, in Task 3, one of the frames was mis-located 3% of the time (Figure 3.19 (c)). Error 2 occurred more often in all 3 tasks. In Task 1, the reproduced path failed to start from the circuit board. This occurred when the circuit board is more offset from the centre of the workspace (Figure 3.19 (a)). In Task 2, the path failed to reach the hand when the ox and hand where too far away from each other (Figure 3.19 (b)). In Task 4, the reproduced path failed to reach the inside of the box which counted as an Error 2 (Figure 3.19 (e)). This occurred when the box was either too close or too far from the cube. In both of these cases, the error is caused by the performance of the TP-GMR. This encouraged the work in chapter 4, which was to adjust the probability distribution used in TP-GMR to better accommodate such tasks. In Task 3, the reproduced path deviated outside the book's outline (Figure 3.19 (d)). This was because one of the frames was detected at an offset angle.

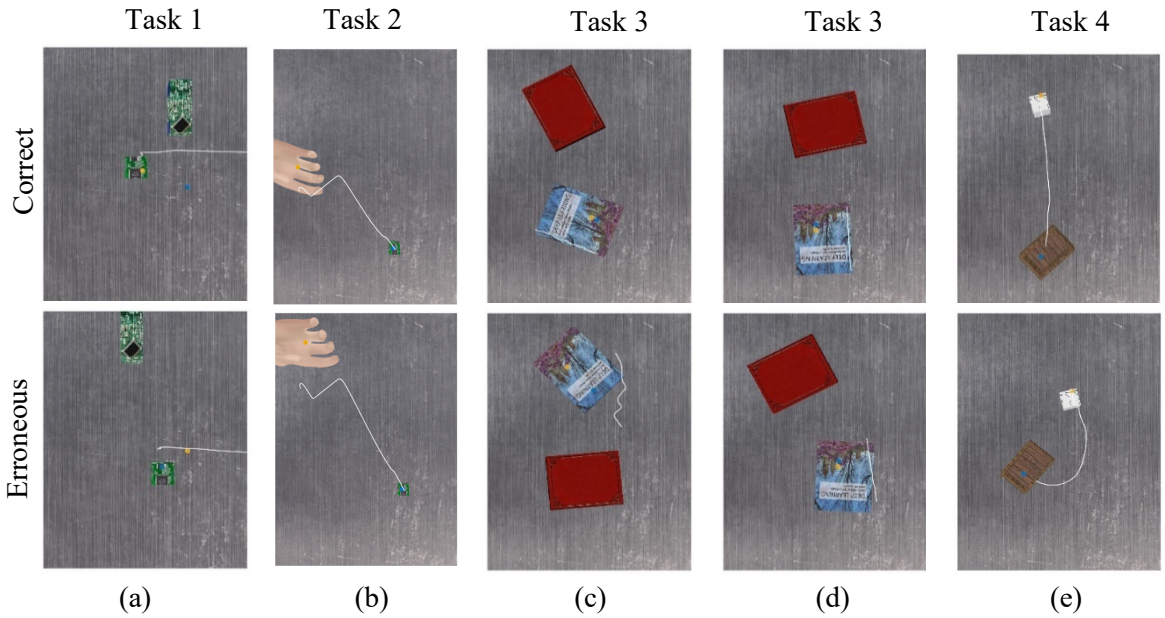


Figure 3.19 Examples of the most common errors in the path reproductions: (a) In Task 1, if the frame belonging to the small circuit board is falsely detected to be on the table, the path will not reach the board; (b) In Task 2, if the hand is too far in the operation space, the reproduced path might fail to reach it; (c) In Task 3, if one of the frames belonging to the book is not detected in its location, the reproduced path will be faulty; (d) In Task 3, if one of the frames is detected at a wrong orientation, the reproduced path will be rotated; (e) In Task 4, when the box is too close to the cube and not facing it from the side, the path will twist.

Finally, when running the algorithm on new images, the average time it took to identify the location of the relevant task parameters and reproduce the path was around 0.55s. The calculations were done on MATLAB on a MacBook Air M1 chip that has a 7-core GPU. Depending on the task, such processing time has the potential to enable real-time performance.

3.4. Conclusion

The task parameters chosen play a very important role in the quality of the reproduced path by task parametrized Gaussian mixture models (TP-GMM) and regression (TP-GMR). This chapter presents a novel algorithm, GenLfD, which automatically detects and identifies the optimal task parameters for learning a given task. The algorithm presented is generic, i.e. can be used to program a wide range of tasks without applying any programmatic changes. Moreover, the algorithm is end-to-end as it requires no inputs from the user except the task demonstrations (path data and setup image).

GenLfD consists of several research innovations: 1) detecting generic visual features from demonstration images to serve as potential task parameters, 2) grouping features that belong to the same objects together to avoid redundancies, and 3) identifying the optimal features to be used as task parameters to learn a given task.

GenLfD was used to successfully and easily program a cobot for four different tasks: applying glue on book spine, pick-and-place, sorting and handover. The results show that GenLfD is a robust algorithm able to learn a variety of tasks involving different objects. Moreover, GenLfD generates reliable results given setup changes. However, there still is room for future improvements of GenLfD, mainly in the following aspects:

- Using a 3D visual feature detector instead of a 2D visual feature detector (SURF), to support more complex shapes and textures of objects
- Learning gripper 3D orientation in addition to gripper 3D position and state, to support more complex tasks
- Detecting visual features in each time step instead of just at the beginning, to support dynamic changes in tasks
- Automatically identifying the optimal number of relevant frames
- Using a smarter cost function that encourages a path as close as possible to the demonstration path, only where it matters (i.e. when the demonstration path variance is low)

Moreover, in light of ISO 10218-1/2:2011, the work in this chapter can encompass the four collaborative safety operative modes in the ISO 10218-1/2:2011 standards: safety-rated monitored stop (SMS), hand-guiding (HG), speed and separation monitoring (SSM) and power and force limiting (PFL). For example, a safety algorithm can be overlayed over GenLfD such that the robot halts (SMS) or varies speed (SSM) if a human gets too close. Moreover, based on the variances learnt through TP-GMM, the robot can vary its stiffness where the path is allowed to vary allowing for the human to guide the robot (HG). Finally, the robot can also limit its power and force to comply with the PFL mode. Therefore, the work in this chapter does not contradict or limit any additional measures that need to be taken in compliance with ISO 10218-1/2:2011.

To conclude, GenLfD enables operator with little programming experience to teach cobots to perform flexible industrial tasks easily and quickly. The cobot learning algorithm, GenLfD, developed in this chapter has great potential to boost the extent and range of use of cobots on factory floors.

Chapter 4: Ring Gaussians for Orientation-less Frames

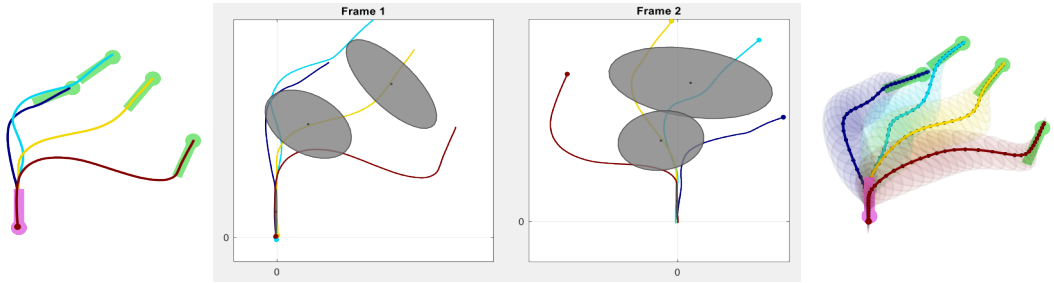
4.1. Introduction

In this chapter, the second thesis contribution is presented. The problem of Gaussian mixture models (GMM) not catering for orientation-less frames is defined. A novel model, called the ring Gaussian, is proposed, and integrated into the task parametrized Gaussian mixture model algorithm. Using this model instead of the GMM, more efficient and successful paths are reproduced when one of the task parameters is an orientation-less frame.

4.1.1. Background

In task parametrized learning from demonstration, various probability distributions can be used to model the path data with respect to the task parameters. In this research, task parametrized Gaussian mixture models (TP-GMMs) are used since the central limit theorem states that normal distributions successfully models many complex systems with the least amount of prior knowledge (Goodfellow et al., 2016). Moreover, the path is reproduced using task parametrized Gaussian mixture regression (TP-GMR).

However, the GMM modelling sets task limitations on the use of TP-GMM. When introducing TP-GMM, Calinon presented the following illustrative example: a path that goes from one point, i.e. frame, to another (Calinon, 2016). However, the path is subject to a constrained slit at each frame (Figure 4.1 (a)). Meaning, the path approaches (or leaves) a frame from a fixed relative orientation. Due to this constraint, data points from various demonstrations overlap for a small portion of the path. When encoding the demonstration paths with a GMM with respect to a frame, a low variance Gaussian will model the path in its slit (Figure 4.1 (b)). When multiplying Gaussians from different frames together, the low variance Gaussians will ensure that the reproduced path satisfies the task conditions of passing through the slit and reaching a frame (Figure 4.1 (c)).



(a) (b) (c)

Figure 4.1 The illustrative example presented by Calinon to describe TP-GMM and TP-GMR. (a) the four demonstrations provided. (b) the GMM modelling the paths observed from each frame of reference. (c) the paths reproduced using the GMMs.

4.1.2. Problem Statement

This is analogous to the dustpan in the cleaning task example presented in the introduction. The path always approaches the dustpan from a specific relative orientation (Figure 4.2 (a)). We defined the frame associated with the dustpan object to be an oriented frame (OF). OFs are frames whose orientations are relevant to a demonstration. If the orientation of the dustpan changes, the demonstration path has to change as well (Figure 4.2 (c)).

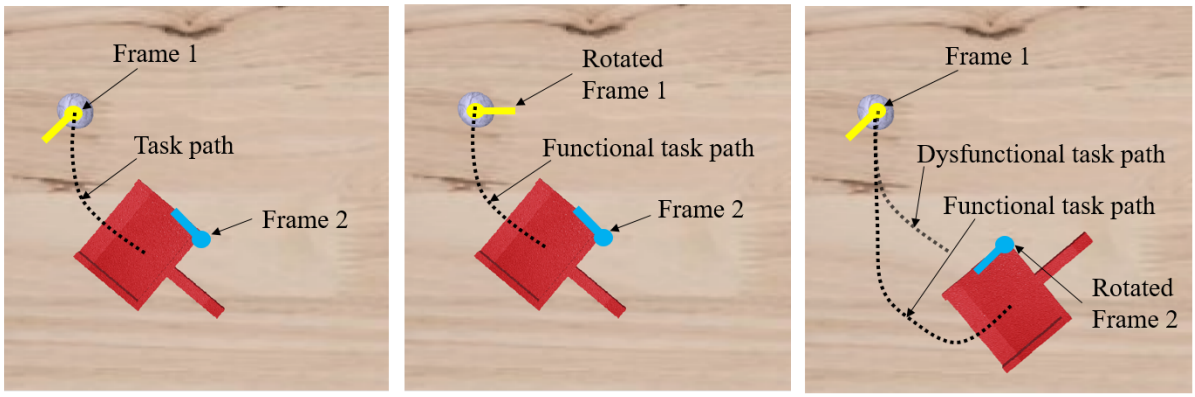


Figure 4.2 The cleaning task example illustrating the difference between OFs and OLFs. (a) The original demonstration path, (b) If Frame 1 is rotated, the original demonstration path is still functional. Thus, Frame 1 is considered an OLF, as its orientation does not play a role in the functionality of the path, (c) If Frame 2 is rotated, the original path becomes dysfunctional, making it necessary to record/generate another updated path. This makes Frame 2 an OF, as its orientation plays a role in the functionality of the path.

However, consider the case of the debris in the cleaning task example presented in the introduction. The path can approach and carry the debris from any relative orientation. In the debris rotates, the path does not need to adjust to accomplish the task (Figure 4.2 (b)). The Gaussian components of the debris' GMM all have high variance due to the absence of a constrained path portion. Therefore, when Gaussians from various frames are multiplied together, the debris' Gaussians will not dominate, and hence the path will fail to reach the debris. We defined the frame associated with the debris object to be an orientation-less frame (OLF).

4.1.3. Relevant Works

Various research works were conducted to improve the performance of the conventional TP-GMM/R. Some researchers undertook demonstration-based improvements. For example, Hu and Kuchenbecker designed TP-GMM/R to program collaborative object movement but iteratively suggested adding demonstrations to improve performance (S. Hu & Kuchenbecker, 2019). Cao et al. used GMM/R iteratively to program robotic motions (Zhiqi Cao et al., 2019). After each GMR, if a collision occurs, a human operator corrects the path and retrains GMM. Willibald developed an interactive learning system in which a robot automatically detects new tasks that require new demonstrations (Willibald, 2020). However, these demonstration-based methods are not a suitable

solution for improving the performance of TP-GMM/R in the case of OLFs. Such methods will still require a human operator to record demonstrations in which paths vary in all directions around an OLF. Even though a set of demonstrations will fully describe the possible paths, the resultant GMM that models these demonstrations could still fail. That is, the GMM will either be biased towards one direction of the frame, or will be centred on the frame, both of which are not accurate representations of the paths.

Other researchers worked on frame-based solutions to improve the performance of TP-GMM/R. For example, Sena et al. calculated the importance scores of frames to amplify the weights of relevant frames during a portion of a demonstration (Sena et al., 2019). Based on that, the TP-GMM/R algorithm can provide meaningful results even when the newly observed positions of frames are far from the previous demonstrations. Vidaković et al. designed an algorithm that classified task parameters/frames (Vidaković et al., 2020). Some frames are classified as attractors and some as obstacles. A cost function was designed to ensure that obstacles are avoided and attractors are approached. Moreover, the cost function can also maintain a specific relative orientation of path points with respect to other frames. Silverio et al. designed new task parameters that help identify positional and/or orientation constraints as well as configurational constraints (Silverio et al., 2019). Instead of considering task parameters to be positions of objects in space, task parameters are Jacobians describing the absolute pose of the bimanual humanoid's end effector as well as their pose relative to the humanoid. These different task parameters are projected on to the configurational joint space of the humanoid. This allows the robot to learn complicated tasks that automatically toggle between positional/orientation and absolute/relative constraints of its end effector. Their work, however, neglects information about the positions of other objects in the same space so that such solution does not tackle the problem of OLFs from its root cause.

To tackle the TP-GMM/R's problems related to OLFs, an OLF requires a different Gaussian model than the conventional one to accurately describe the nature of demonstration paths. None of the previous works to the best of our knowledge, tackled this problem from the TP-GMM/R-based point of view. In this chapter, we create a new Gaussian model that accurately encodes demonstration paths with respect to OLFs. OLFs are automatically identified in a task. Their corresponding new Gaussian model, called ring Gaussian, is automatically calculated and used to reproduce the path in a new task setting. The overall algorithm is referred to as ring TP-GMM/R.

4.2. Methodology

The overall framework entails the following steps: 1) Collecting key information in demonstrations, i.e., task parameters (frames) and paths (refer to Section 3.1). A user provides demonstrations by dragging the end effector of a cobot to create some trajectories from the start position to the end position of a task. 2) Training TP-GMM based on demonstrations (refer to Section 3.2). 3) Identifying OLFs and OFs. 4) Calculating ring Gaussians for the OLFs. 5) Upon being given new data for task parameters, converting the ring Gaussians of OLFs to Gaussians. 6) Performing the TP-GMR algorithm to

reproduce a path in a new scenario (refer to Section 3.3). The ring TP-GMM/R algorithm differs over the "conventional" TP-GMM/R in Steps 3, 4, and 5, which are further elaborated in this section. Figure 4.3 illustrates the complete steps of the algorithms. To simplify representations and formulas, in this chapter, demonstrations are presented in a two-dimensional Euclidean space for better explanation and illustration. The developed algorithms can be naturally extended to the three-dimensional Euclidean space.

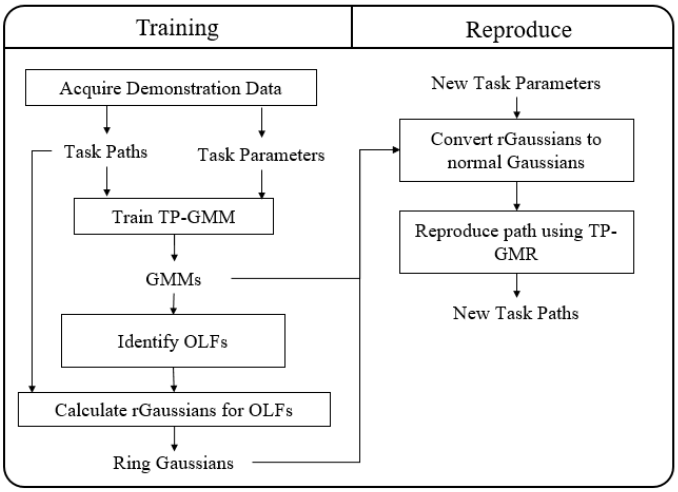


Figure 4.3 The overall framework of the ring TP-GMM/R algorithm.

4.2.1. Identifying Orientation-less Frames

As shown in Figure 4.3, identifying OLFs is performed after training the TP-GMM model. The TP-GMM assumes all frames are OFs. The GMMs encoding the demonstration paths with respect to each frame are obtained. Identifying OLFs is subsequently done by identifying OFs. OFs are distinguished by the overlapping portion of the demonstration paths, which is modelled using a Gaussian component of low variance, as in Figure 4.1. Such a Gaussian component has three distinguishable criteria. Once all three criteria are ensured, a frame is labelled as an OF. Otherwise it is an OLF.

The three criteria are:

The Gaussian should encode points from all demonstrations

TP-GMM is designed to model patterns and trends in all demonstrations. That is, each resultant Gaussian needs to provide information about the patterns and trends of all demonstrations. However, one of the limitations of TP-GMM is that it does not have knowledge of which points belong to which demonstration. That is, all points are regarded equally during TP-GMM training so that the algorithm might generate Gaussians that are biased towards certain demonstrations over others. In some cases, it might even make a Gaussian to over-fit a demonstration, as depicted in Figure 4.4 (a).

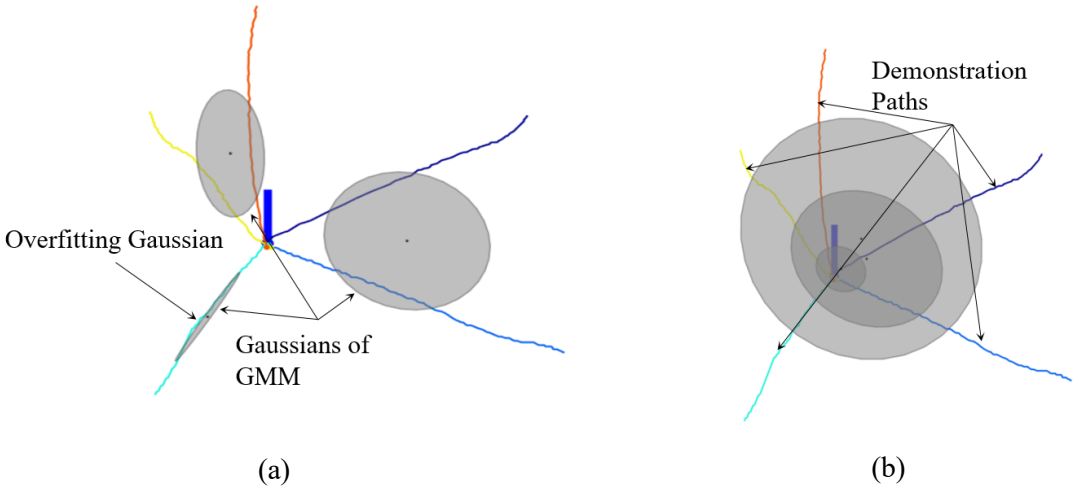


Figure 4.4 Result of TP-GMM: (a) depicting an anomaly Gaussian over-fitting one demonstration, (b) in which over-fitting is avoided.

As described in chapter 2, TP-GMM is trained using the EM algorithm. In every E-step, the likelihood $\mathcal{N}\left(\boldsymbol{X}_t^p|\mu_{p,k},\Sigma_{p,k}\right)$ of each point $\boldsymbol{X}_t^p$ at time step t belonging to a Gaussian $\mathcal{N}\left(\mu_{p,k},\Sigma_{p,k}\right)$ is calculated. We recall that $\boldsymbol{X}_t^p$ is a concatenation of points $X_t^{p,m}$ from all demonstrations $m \in \{1\dots M\}$. The calculation is conducted independently for each point, which means that the EM algorithm does not account for which demonstration that point belongs to. This potentially allows Gaussians to over-fit a particular demonstration. Based on this observation, it is essential to introduce adjustments to the EM training algorithm to ensure each obtained Gaussian models points from all demonstrations equally, to avoid such anomalies. The following steps are followed:

1. Calculate the likelihood $\mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right)$ of a point $X_t^{p,m}$ for all values of m.
2. For each Gaussian component k, let L_{tk} be the average of $\mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right)$ across all demonstrations. This step associates the likelihoods of the points of the time step t together.

$$L_{tk} = \frac{1}{M}\sum_{m=1}^{M}\mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right) \tag{4.1}$$

In reality, point $X_t^{p,m}$ of demonstration m might not be associated with point $X_t^{p,m'}$ of Demonstration m'. Demonstrations might have slightly varying lengths and velocities, thus some points from different demonstrations that are closer in the x-y space might have slightly different time steps t. Therefore, the values of L_{tk} are "blended" across t by using weighted average.

3. Define a kernel of size 2S+1 such that,

$$kernel = [f_1, f_2, \dots, f_{2S}, f_{2S+1}]\ where\ f_{S+1-s} = f_{S+1+s} = 1 - \frac{s}{S+1}\ for\ s \in [0, \dots, S] \tag{4.2}$$

For example, if S=3, *kernel* = [0.25, 0.5, 0.75, 1, 0.75, 0.5, 0.25].

4. Pad L_{tk} with S zeros on each side of dimension t, such that the size of L_{tk} is now $\{T+2S, K\}$.
5. For each Gaussian component k, convolve the *kernel* over the vector L_{tk} across t.

$$L_{tk} = \frac{1}{1+S}\left(\sum_{index=1}^{2S+1} kernel_{index} \times L_{(t-S-1+index,k)}\right) \tag{4.3}$$

6. Adjust the likelihood $\mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right)$ by bringing it closer to L_{tk} by a scale ω. If ω=1, then $\mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right) = L_{tk}$. If ω = 0, the values $\mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right)$ would be left unaffected. Therefore, an intermediate value of $\omega = 0.5$, prevents over-fitting without enforcing rigid constraints.

$$\mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right) = \mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right) + \omega\left(L_{tk} - \mathcal{N}\left(X_t^{p,m}|\mu_{p,k},\Sigma_{p,k}\right)\right) \quad (4.4)$$

Thus, the resultant Gaussians have a tendency to model all demonstrations evenly, instead of over-fitting towards one particular demonstration, such as in Figure 4.4 (b).

The Gaussian should be narrow

A Gaussian is characterised by its Eigen vectors [v_1, v_2] and Eigen values [e_1, e_2], which describe the Gaussian's direction and size, respectively (Figure 4.5).

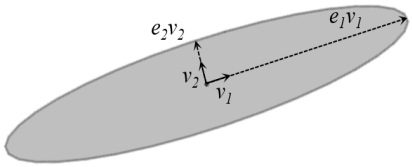


Figure 4.5 The Eigen values and Eigen vectors of a Gaussian.

Assuming a 2 dimensional Gaussian for x-y coordinates, let Σ_{xy} be its covariance matrix. The Eigen values and vectors are calculated using the following equations:

$$\Sigma_{xy} \cdot v_1 = e_1 \cdot v_1 \quad (4.5)$$

$$\Sigma_{xy} \cdot v_2 = e_2 \cdot v_2 \quad (4.6)$$

To consider a Gaussian to be narrow, the ratio $r = e_1/e_2$ should be greater than a threshold ε. ε is chosen to be 10 by trial and error. A narrow Gaussian indicates that the portions of demonstrations that it encodes are in restricted or limited variability with respect to its corresponding frame. This criterion, when combined with the following criteria, helps identify whether the orientation of that frame affects demonstrations, i.e. if the frame is an OF.

The Gaussian should stretch along the direction of demonstration paths

This criterion ensures that the Gaussian is not stretched over demonstration paths that are widely apart such as illustrated in Figure 4.6 (a). Instead, the direction of change of time step *t* (signifying the direction of demonstrations) should be parallel to the long axis of the Gaussian, such as illustrated in Figure 4.6(b).

The direction of change of time step *t* is given by the vector $C_t = (C_{xt}, C_{yt})$, where C_{xt} and C_{yt} are components from the covariance matrix $\Sigma_{p,k}$ of the Gaussian *k*.

$$\Sigma_{p,k} = \begin{bmatrix} C_{tt} & C_{xt} & C_{yt} \\ C_{tx} & C_{xx} & C_{yx} \\ C_{ty} & C_{xy} & C_{yy} \end{bmatrix} \quad (4.7)$$

C_{xt} describes the effect of the *x* position of a point on its time step *t*. The change in time step *t* signifies the flow on a demonstration, since the points increase time step *t* incrementally along the

demonstration. That is, the higher C_{xt} value is, the more the change in the x coordinate affects the change in time step t. That means that demonstrations are more tending to be parallel to the x axis. Similarly, the same applies for C_{yt}. Therefore, the vector $C_t = (C_{xt}, C_{yt})$ is indicative of the direction of the flow of demonstrations in the x-y space.

The direction of the long axis of the Gaussian is given by Eigen vector $\boldsymbol{v_1}$ given that Eigen value e_1 is greater than Eigen value e_2. Angle α between these two vectors C_t and v_1 is calculated using their dot product. To comply with the criterion, trials show that the angle α must be equal to 0 ± 0.1 or $\pi \pm 0.1$.

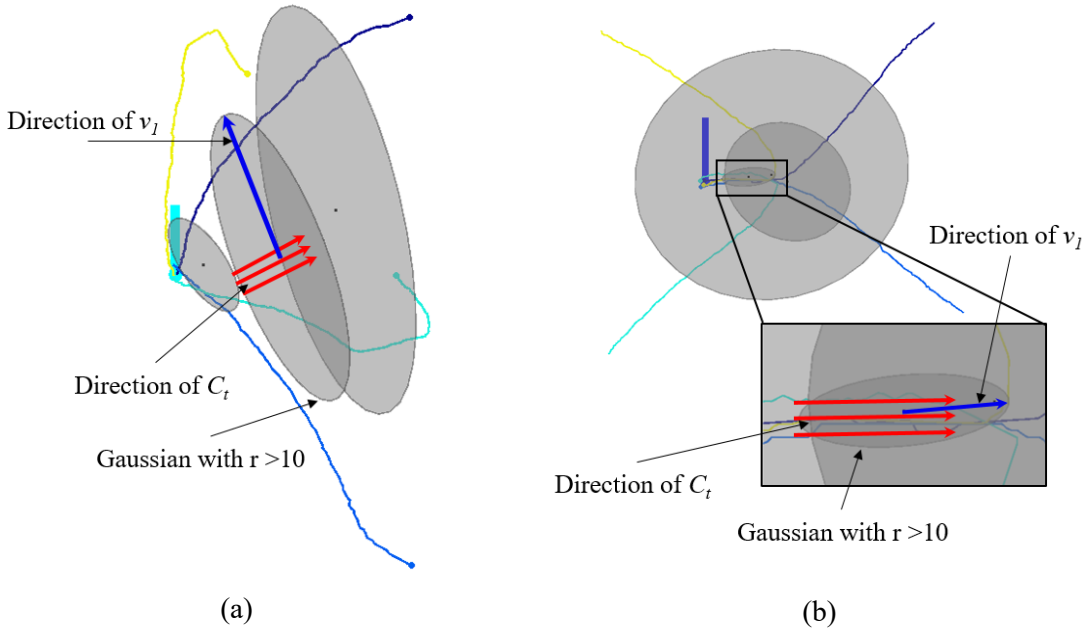


Figure 4.6 An example of a narrow Gaussian that: (a) does not belong to an OF. The diagram shows that the direction of the change of time step t is not parallel to that of the Gaussian's long axis, (b) belongs to an OF. The diagram shows that the direction of the change of time t is almost parallel to that of the Gaussian's long axis

Therefore, if a Gaussian fulfils the above three criteria, it is said the frame that the Gaussian is associated with is an OF. Otherwise, the frame is an OLF.

4.2.2. Calculating Ring Gaussians

No changes are made to the trained TP-GMM if frames are OFs. However, for OLFs, new probability distributions are generated to model the demonstrations. The points of demonstrations around OLFs are modelled in a new probabilistic distribution, as opposed to the "normal" Gaussian used in the "conventional" TP-GMM/R algorithm. Since the frame is orientation-less, a given path point is equally likely to occur around the frame. Therefore, a suitable Gaussian to describe the distribution of points around an OLF would be an *ring* Gaussian, which is described as a Gaussian that has been spanned around the OLF forming a ring (illustrated in Figure 4.7).

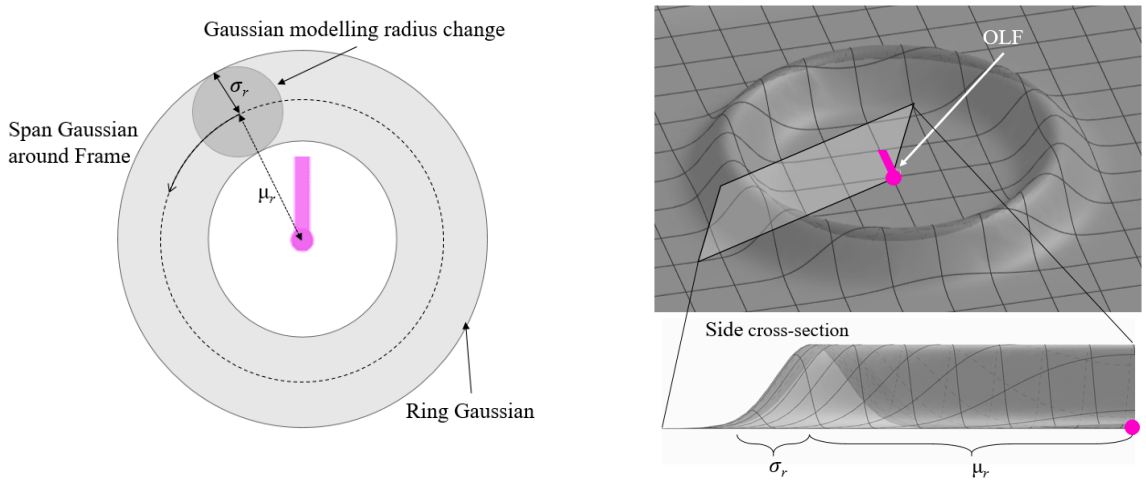


Figure 4.7 The ring Gaussian modelling around an OLF.

A *ring* Gaussian is computed below in a 2 dimensional x-y coordinate system. For every time step *t*, the probability of a path point with respect to the frame is described below:

$$Pr(x,y) = \frac{1}{\sqrt{2\pi{\sigma_r}^2}} e^{-\frac{\left(\sqrt{x^2+y^2}-\mu_r\right)^2}{2{\sigma_r}^2}} \tag{4.8}$$

where $\{x,y\}$ are the 2D coordinates of the path point relative to the frame, μ_r is the mean distance between the point and the frame, and σ_r is the standard deviation of the distance between the point and the frame.

To obtain μ_r and σ_r of every time step *t,* a TP-GMM that models radius *r* and time steps *t* is trained. For example, take OLF *p* in Figure 4.8 and five demonstration paths (M=5) around it. As previously mentioned, each demonstration constitutes of *T* path points represented in time step *t* and the *x*-*y* position coordinates with respect to Frame *p*. To generate the parameters for the *ring* Gaussian for Frame *p* at time step *t*, the following steps are followed:

1. The *x*-*y* coordinates of the path points with respect to Frame *p* are converted into polar coordinates. That is, the radius of
2. a path point at time step *t* of Demonstration *m* is calculated using the following equation:

$$r_{t,p}^{m} = \sqrt{\left(x_{t,p}^{m}\right)^2 + \left(y_{t,p}^{m}\right)^2} \tag{4.9}$$

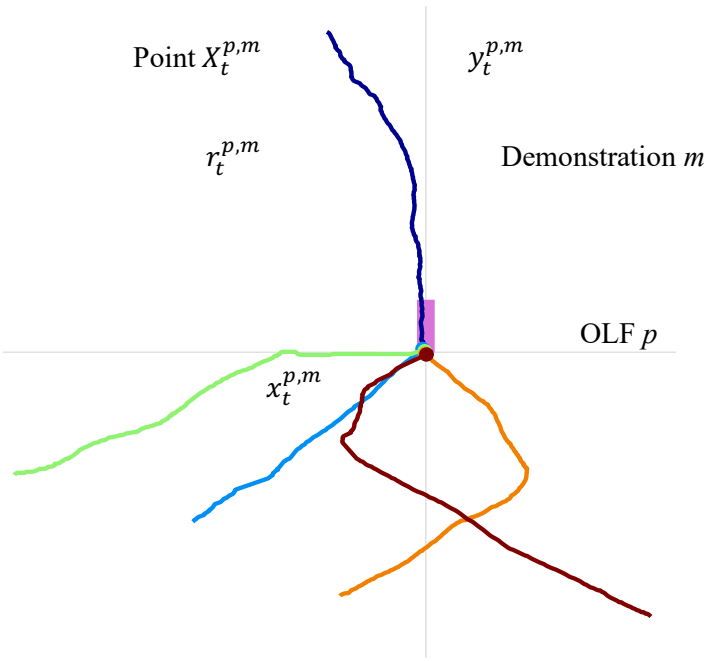


Figure 4.8 An OLF p and five demonstration paths. Each point X on a path has x-y coordinates with respect to Frame p. Moreover, it has a time step t depending on its sequence order in its path. The radius r is calculated as the distance between the frame and the point.

3. A GMM is trained on the 2D data $\{t, r_{t,p}^m\}$ for all $t \in [1,\ldots, T]$ and $m \in [1,\ldots, M]$. This means that K Gaussians are fitted to describe the change in the distance between OLF p and the path points around it. Gaussian k is identified as $\mathcal{N}_{(r)}\left(\mu_{(r)p,k}, \Sigma_{(r)p,k}\right)$. For example, in Figure 4.9, K=3.

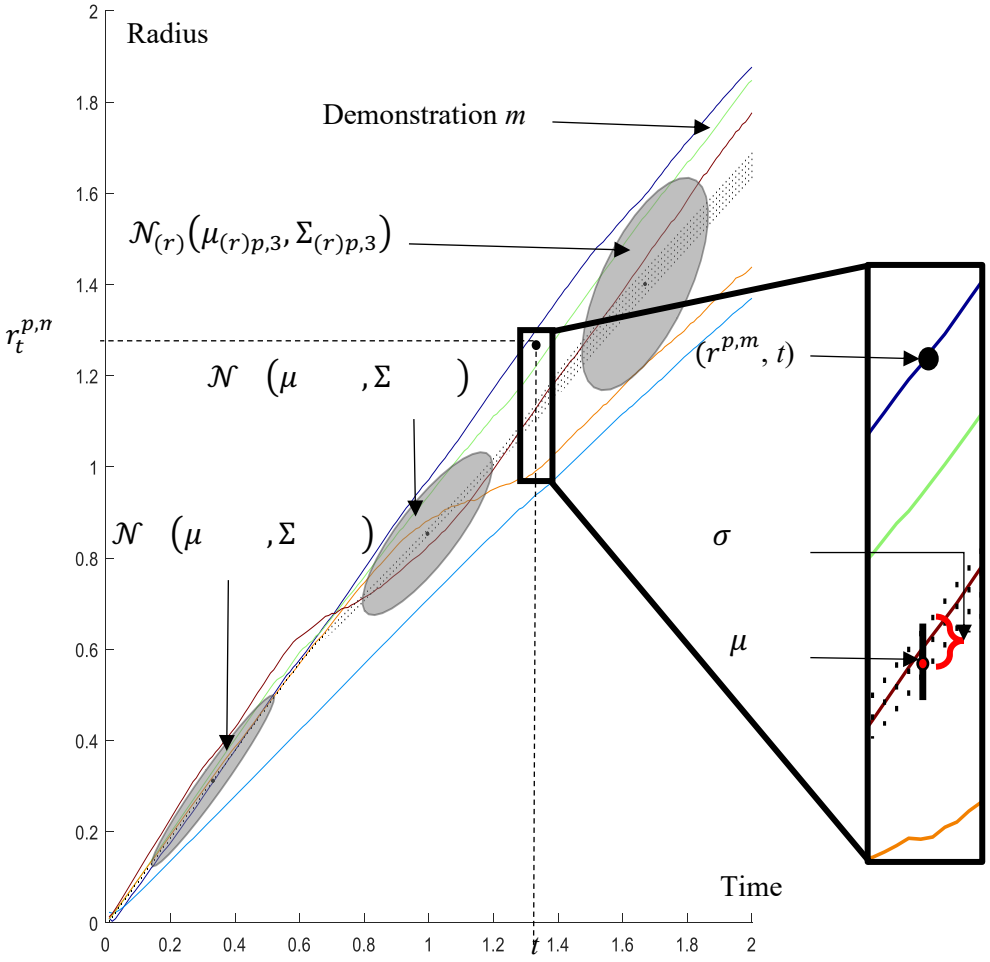


Figure 4.9 Given radius values of all points from the demonstrations, a GMM is fitted to the radius-time data. Then, regression is performed to obtain a mean $\mu_{(r)p,t}$ and a standard deviation $\sigma_{(r)p,t}$ for each time t

4. Gaussian regression is performed to obtain a radius mean $\mu_{(r)p,t}$ and standard deviation $\sigma_{(r)p,t}$ at every time step t (Figure 4.9).

Finally, the resultant data, average $\mu_{(r)p,t}$ and standard deviation $\sigma_{(r)p,t}$ for every time step t, is used in the adjusted TP-GMR in the next section to reproduce paths in new settings.

4.2.3. Reproducing the Path

In a new task setting, the objects are expected to be in new positions that are previously unseen by the cobot. The cobot needs to perform the task path based on what was observed in the demonstrations. The path is reproduced using TP-GMR. Firstly, in TP-GMR, the GMM of each task parameter are regressed from K Gaussians to T Gaussians, i.e. a Gaussian for each time step t. Secondly, the Gaussians from different task parameters are combined together using weighed product (for more details, refer to chapter 2). Weighted product can only be performed on normal Gaussians due to their mathematical properties. That is, only two normal Gaussians can be multiplied together to obtain another normal Gaussian. Therefore, it is imperative that the ring Gaussians of OLFs are converted to normal Gaussians

before performing TP-GMR. To convert a ring Gaussian for OLF p at time step t, the following steps are taken:

1. For all p' ∈ [1,…, P] – {p}, identify the closest point $point_{pp',t}$ on the circle of radius $\mu_{(r)p,t}$ and centre frame p, and
    a. The circle of radius $\mu_{(r)p',t}$ and centre frame p' if frame p' is an OLF. This can be divided into three cases depending on the relative positions of the two circles: 1) The circles are external, i.e. the sum of radii is less that the distance between their centres. 2) The circles are intersecting, i.e. the sum of radii is more than the distance between their centres but the difference between the radii is less than the distance between their centres. 3) The circles are internal, i.e. one of them is inside the other such that the difference between the radii is more than the distance between their centres. Figure 4.10 shows an example of where $point_{pp',t}$ would be for different circle relative positions.

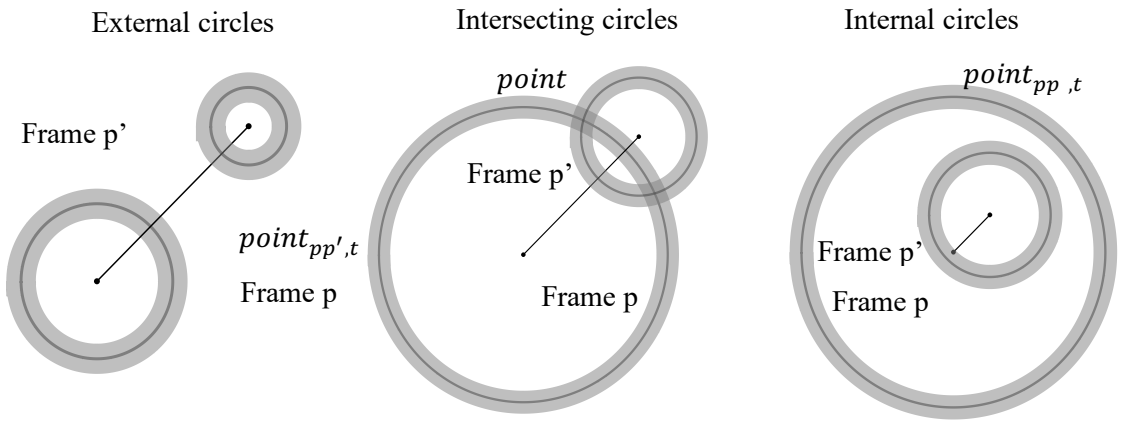


Figure 4.10 The position of $point_{pp',t}$ according to the different relative positions between circles of centres frames p and p' and radii $\mu_{(r)p,t}$ and $\mu_{(r)p',t}$

    b. The centre $\mu_t^{\mathcal{O}}$ of the Gaussian for frame p' for all p' ∈ [1,…, P] – {p} if frame p' is an OF
2. For all p' ∈ [1,…, P] – {p}, calculate the coefficient $d_{pp',t}$ for $point_{pp',t}$ such that:
    a. $d_{pp',t}$, when frame p' is an OLF, is: 1) the distance between $point_{pp',t}$ and the circle of radius $\mu_{(r)p',t}$ and centre frame p' if the circles are external; 2) number very close to zero when the circles are intersecting. Here, we set it to be e-20; 3) the distance between $point_{pp',t}$ and the circle of radius $\mu_{(r)p',t}$ and centre frame p' when the circles are internal.
    b. $d_{pp',t}$ is the distance between $point_{pp',t}$ and the centre $\mu_t^{\mathcal{O}}$ of the Gaussian for frame p' if frame p' is an OF

3. Calculate the weighted average $point_{p,t}$ of points $point_{pp',t}$ for all p' ∈ [1,…, P] – {p}, weighted by $d_{pp',t}$
4. Define the normal Gaussian with centre $point_{p,t}$ and covariance matrix $\Sigma_{p,t}$, such that

$$\Sigma_{p,t} = \begin{bmatrix} \sigma_{(r)p,t} & 0 \\ 0 & \sigma_{(r)p,t} \end{bmatrix} \tag{4.10}$$

Upon converting all ring Gaussians for all OLFs and time steps t to normal Gaussians, TP-GMR can be performed as described in chapter 2, to reproduce the task path.

4.3. Results

4.3.1. Assessing the Performance of the OLF Identifier

The performance of the OLF identifier depends on two main factors: 1) the quality and variability of demonstrations provided, and 2) the ratio between *K,* the number of Gaussians in the GMM, and the length of the orientation-consistent path. An orientation-consistent path is the portion of demonstration paths that is dependent on the frame's orientation. To assess the performance of the OLF identifier, its F1-score is calculated as the following parameters are varied:

- Number of Gaussian components in the GMM (*K*): {3, 4}
- Number of demonstrations used in training (*M*): {4, 5, 6}
- Length of the orientation-consistent path (*L*): {10, 15, 20, 25}% of the average distance between the frames

For this experiment, 6 demonstrations were recorded for a path going from Frame 1, an OF, to Frame 2, an OLF. The algorithm should classify each of the frames correctly as OLF or OFs. The F1-score is calculated as follows:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4.11}$$

where,

$$Precision = \frac{nb\ of\ OLFs\ identified\ as\ OLFs}{nb\ of\ frames\ identified\ as\ OLFs} \tag{4.12}$$

$$Recall = \frac{nb\ of\ OLFs\ identified\ as\ OLFs}{nb\ of\ OLFs} \tag{4.13}$$

Table 4.1 F1 scores for 30 different runs of the OLF identifier with varied parameters.

	M=4, *K*=3	*M*=5, *K*=3	*M*=6, *K*=3	*M*=4, *K*=4	*M*=5, *K*=4	*M*=6, *K*=4
L=10%	0.67	0.67	1	1	0.67	1
L=15%	1	1	1	1	1	1
L=20%	1	0.67	1	1	1	1
L=25%	1	1	1	1	1	1

In Table 4.1, an F1 score of 1 indicates that Frame 1 and Frame 2 were correctly identified as OF and OLF, respectively. A result of 0.67 indicates that the OLF failed to be identified as an OLF. The results show a correlation between the three different parameters and the effectiveness of the OLF

identifier. Based on these correlations, the following recommendations are provided for a user to choose parameters that increase the success chances of the OLF identification:

- Identifying false OLFs can happen when L is low compared to the number of Gaussians K. During the initialisation process of TP-GMM, Gaussians are distributed equally across time segments. For example, if K=4, the first Gaussian initially covers 25% of the path. Whereas for K=3, the first Gaussian covers 33.3% of the path. During TP-GMM training, Gaussians shift across time segments in order to accurately model the patterns and trends of the path. For example, when L=15%, ideally after convergence, the first Gaussian should model the first 15% of the path in order to accurately describe it. Correctly describing it means a Gaussian will conform to the criteria mentioned in Section 3.1, thus increasing the chances of identifying the frame as an OLF. Therefore, when L is much less than 100/K, there seems to be a higher chance of identification error.

  Usually, the number of Gaussians K is optimised using Bayesian Information Criterion (BIC) (Schwarz, 1978) or by calculating a task-relevant cost function for different value of K and choosing the one with the least cost (Rozo et al., 2016). Alternatively, the value of K might be manually selected (Hewitt et al., 2017). However, the mentioned results encourage to consider the length L, which is determined or restricted by the task objects and the task requirements. Moreover, the length L is encouraged to be if possible, given that it doesn't hinder the functionality or constraints of the task. This should be judged by the operator recording the demonstrations.
- Increasing the number of demonstrations results in a more varied set of demonstrations, which in turn results in a more accurate and well fitted GMM. This decreases the OLF identification false positives.

To further understand the effect of the kernel convolution (Section 4.2.1.) that was designed to avoid overfitting, the OLF identifier was executed without it. In Table 4.2, an F1 score of Inf indicates that Frame 2 was mistakenly identified as an OF. That is because when the kernel convolution is not performed, some Gaussians might over fit a particular demonstration's path thus fulfilling two of the OLF identifier's criteria: narrow Gaussian and Gaussian that stretches along the path direction. Therefore, the kernel convolution plays an important role in preventing this error.

Table 4.2 F1 scores for 30 different runs of the OLF identifier with varied parameters, without convolving the overfitting kernel.

	M=4, K=3	M=5, K=3	M=6, K=3	M=4, K=4	M=5, K=4	M=6, K=4
L=10%	Inf	1	1	Inf	1	Inf
L=15%	1	1	1	1	1	1
L=20%	1	1	1	1	1	1
L=25%	Inf	1	1	Inf	1	1

4.3.2. Synthetic Data

The ring TP-GMM/R algorithm was tested on 3 synthetic tasks, similar to the standard task presented by Calinon in (Calinon, 2016). The task is to go from Frame 1 (pink) to Frame 2 (green), under varying conditions:

Task 1: Both frames are oriented (OFs). Therefore, the path has to pass through Frame 1 and Frame 2's tips as seen in Figure 4.11. The training parameters were: M=4, K=3. This is analogous to an assembly task.

Task 2: Frame 1 is oriented (OF) while Frame 2 is orientation-less (OLF). Therefore, the path has to pass through Frame 1's tips but can approach Frame 2 from any direction as seen in Figure 4.11. The training parameters were: M=6, K=4. This is analogous to a peg-in-hole task.

Task 3: Both frames are orientation-less (OLFs). Therefore, the path can approach Frame 1 and Frame 2 from any direction as seen in Figure 4.11. The training parameters were: M=6, K=4. This is analogous to a pick-and-place task.

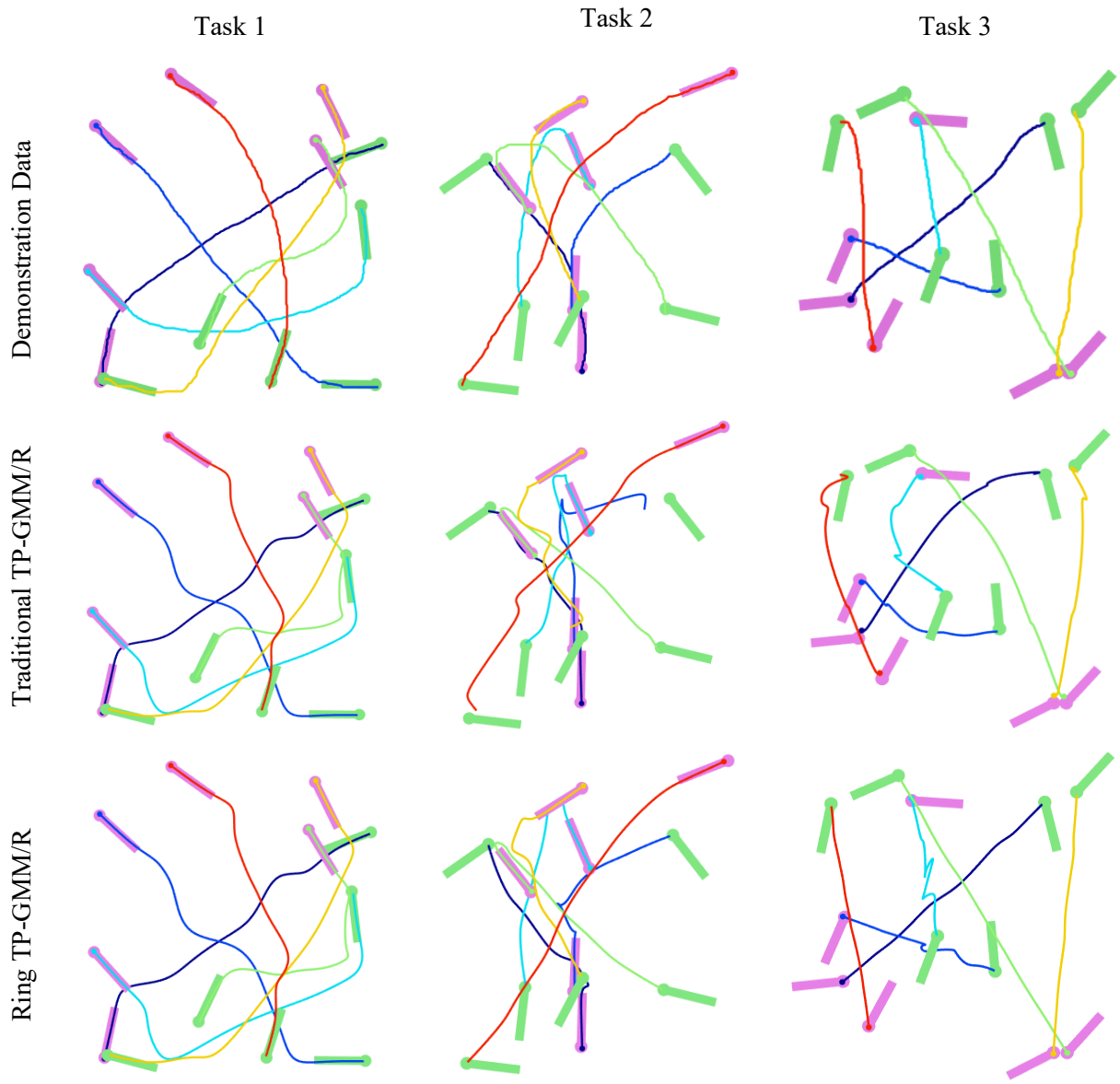


Figure 4.11 The demonstration data for the three different synthetic tasks.

All three demonstrations were learnt using both traditional TP-GMM/R and ring TP-GMM/R. The ring TP-GMM/R automatically identifies which frames are oriented or orientation-less. The reproduced paths are shown in Figure 4.11.

In Task 1, the ring TP-GMM/R successfully identified the frames as oriented and used normal Gaussian to model the demonstration paths. Therefore, the reproduced paths are the identical using both algorithms.

In Task 2, the reproduced path sometimes failed to reach the green Frame 2 using traditional TP-GMM/R since the normal Gaussian modelling does not accurately reflect orientation-less frames. This error was not encountered when using ring TP-GMM/R. Moreover, the ring TP-GMM/R also respected the constrained imposed by the oriented Frame 1, even when an OLF is present in conjunction with an OF.

In Task 3, it can be noticed than the reproduced paths by the improved algorithm are straighter than those by the conventional algorithm.

Overall, the ring TP-GMM/R provides comparably satisfactory results with OFs and better results with OLFs than the traditional TP-GMM/R. A quantitative comparison is performed on a 100 reproductions of the task paths on new previously-unseen positions of the frames. The results are assessed using three metrics: smoothness, efficiency and reachability.

Smoothness. The generated path should ideally be smooth such that there are no sharp turns that might jolt a cobot upon performing the path. Firstly, the derivative $\frac{dy_t}{dx_t}$ of the path at each point $p_t = \{x,y\}_t$ of time step t is calculated. This derivative describes the slope of the path. Then, the second derivative is calculated with respect to time. The second derivative $\frac{d\left(\frac{dy_t}{dx_t}\right)}{dt}$ describes the rate of change in the path's slope, i.e., a high change would signify a sharp edge. For every point at which the second derivative is higher than a threshold (e.g., 0.5), a "sharp" edge is noted. The smoothness score is given based on the average number of sharp edges per path across the 100 reproductions. Therefore, the higher the smoothness score, the more sharp edges there are, so that the less smooth the path is.

$$\frac{dy_t}{dx_t} = \frac{y_{t+1} - y_{t-1}}{x_{t+1} - x_{t-1}} \tag{4.14}$$

$$\frac{d\left(\frac{dy_t}{dx_t}\right)}{dt} = \frac{\left(\frac{dy_{t+1}}{dx_{t+1}} \frac{dy_{t-1}}{dx_{t-1}}\right)}{2} \tag{4.15}$$

$$edges_{m,t} = \begin{cases} 1\,, \; d(\frac{dy_t}{dx_t})/dt > 0.5 \;\; for\, reproduction\, path\, m \in [1, \dots, 100] \\ 0\,, \; d(\frac{dy_t}{dx_t})/dt \leq 0.5 \;\; for\, reproduction\, path\, m \in [1, \dots, 100] \end{cases} \tag{4.16}$$

$$Smoothness = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T} edges_{m,t} \tag{4.17}$$

Efficiency. The path should also be efficient in length, i.e. as close to the shortest distance as possible. The efficiency score is obtained by dividing the shortest distance sd_m between the start and end points by the distance d_m covered by the reproduced path m. Therefore, the closer the score is to 1, the closer the path is to the shortest distance.

$$sd_m = \sqrt{(y_1 - y_T)^2 + (x_1 - x_T)^2} \tag{4.18}$$

$$d_m = \sum_{t=1}^{T-1} \sqrt{(y_t - y_{t+1})^2 + (x_t - x_{t+1})^2} \tag{4.19}$$

$$Efficiency = \frac{1}{M} \sum_{m=1}^{M} \frac{sd_m}{d_m} \tag{4.20}$$

Reachability. The *reachability* shows the percentage of the path successfully passing in the start and end points. The reason to consider this measurement is that even if a reproduced path is smooth and efficient, if it does not achieve successful reachability, the task fails. Firstly, for each reproduction path m, the distance $d1_m$ between the start frame $\{X_1, Y_1\}$ and the start point $\{x_1, y_1\}$ on the path is measured.

Similarly, the distance $d2_m$ between the end frame $\{X_2, Y_2\}$ and the last point $\{x_2, y_2\}$ on the path is measured. For each value of $d1_m$ or $d2_m$ that is less than an error tolerance value ε, the *reachability* score is increased by 0.5. The error tolerance is manually chosen and can change depending on the task and the scale/size of the task objects involved. Having a lesser ε value means that the task is more strict in reaching start and end positions. Finally, the *reachability* score is averaged over the total number of reproductions M.

$$d1_m = \sqrt{(y_1 - Y_1)^2 + (x_1 - X_1)^2} \tag{4.21}$$

$$d2_m = \sqrt{(y_2 - Y_2)^2 + (x_2 - X_2)^2} \tag{4.22}$$

Starting from $Reach = 0$ and iterating over the values of $m \in [1, \dots, M]$:

$$Reach = \begin{cases} Reach, & d1_m > \varepsilon \text{ and } d2_m > \varepsilon \\ Reach + 0.5, & d1_m < \varepsilon \text{ or } d2_m < \varepsilon \\ Reach + 1, & d1_m < \varepsilon \text{ and } d2_m < \varepsilon \end{cases} \tag{4.23}$$

$$Reach = Reach \times 100 / M \tag{4.24}$$

Table 4.3 Metric results on the three different tasks using the traditional TP-GMM/R and the ring TP-GMM/R training algorithms.

Task/Algorithm		Smoothness	Efficiency	Reachability (%)
Task 1	Traditional TP-GMM/R	43.5200	2.2506	99
	Ring TP-GMM/R	43.5200	2.2506	99
Task 2	Traditional TP-GMM/R	46.55	1.499	62.5
	Ring TP-GMM/R	29.95	1.456	100
Task 3	Traditional TP-GMM/R	50.76	1.1123	62.5
	Ring TP-GMM/R	29.78	1.0698	100

The results in Table 4.3 show that the ring TP-GMM/R exceeds the traditional TP-GMM/R in performance in all three metrics when an OLF is included in the task. The paths produced using the ring TP-GMM/R have less sharp edges, reach their targets more efficiently and effectively.

However, this performance is dependent on the following assumption: In the ring Gaussian, the centre of the ring is always the orientation-less frame with respect to which the paths are being modelled. Therefore, when a path starts or ends at a certain frame, the ring Gaussian is capable of capturing that with a ring of a very small radius. However, if a path starts or ends at a point far offset from a frame, then the ring Gaussian will have a radius equal to the distance between the frame and the offset point. However, that means that the path can start or end anywhere on the ring Gaussian, not necessarily at the intersection between the ring Gaussian and the offset point. Moreover, the algorithm assumes the absence of obstacles and that the ideal path from the origin to the destination is a straight line. That excludes any path constraints imposed by oriented frames.

4.3.3. Simulation Results

To assess the results of the algorithm in chapter 4, we perform the experiment on Tasks 1, 2 and 4 from chapter 3. In chapter 3, the orientation of relevant objects was maintained such that all the frames were oriented frames. That was done to highlight the effect of chapter 3's work independently from chapter 4's. However, in this chapter, the orientation of relevant objects is varied such as they are OLFs. The effectiveness of the ring Gaussian at modelling paths with respect to OLFs is examined.

The simulation scenes were built in CoppeliaSim EDU. As a reminder, the tasks performed are the following:

Task 1 - Sorting: Given two circuit boards and two containers, the robot task is to pick the smaller circuit board and place it in the green container. The containers are in a fixed positions where as the circuit boards vary position and orientation. Task 1 shows that ring Gaussians can cater for OLF even if they are fixed in position.

Task 2 - Handover: The cobot needs to pick up an object, in this case a circuit board, and hand it to a human hand. Both the hand and the circuit board vary positions and orientations.

Task 4 - Pick-and-place: The robot is require to pick an object, a cube, and place it in a box. Both the cube and the box vary positions and orientation on a table. This could also be analogous to peg-in-hole or assembly tasks.

Six demonstrations are recorded for each task where the objects are in variable positions. Five of these demonstrations are used for training, and one of them is used for validation. The following parameters were set as defaults during training, like in chapter 3, and were not changed between tasks:

- Number of demonstrations $M = 5$
- Number of Gaussian components $K = 4$
- Number of relevant frames $nbRelev = 2$

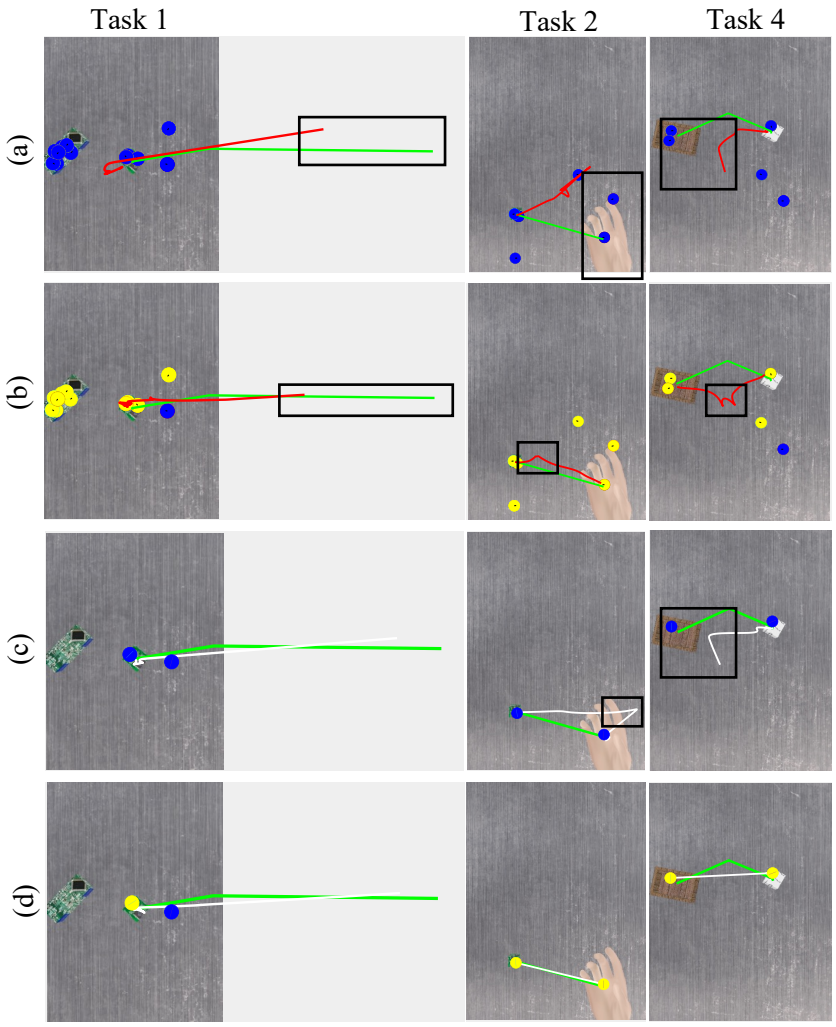


Figure 4.12 The reproduced path for Tasks 1, 2 and 4 when: (a) all frames are used in TP-GMM and considered OFs by default, (b) all frames are used in TP-GMM and identified to be OLFs or OFs, (c) relevant frames are used in TP-GMM and considered OFs by default, and (d) relevant frames are used in TP-GMM and identified to be OLFs or OFs.

Figure 4.12 shows the results of the algorithm at different stages. Row (a) shows the reproduced path (red) compared to the demonstration path (green) when traditional TP-GMM/R is used and all frames are considered oriented (blue) by default. All the reproduced paths failed to reach their destination, both because all frames are oriented by default and because some of them are irrelevant.

In row (b), the frames are identified to be oriented or orientation-less. The images show the reproduced path (red) compared to the demonstration path (green) when ring TP-GMM/R is used and some frames are identified as orientation-less (yellow). An improvement is noticed compared to the paths in the first row. The paths in Task 1 and 2 successfully reach their destination. However, they are unsmooth.

Row (c) shows the reproduced path (white) compared to the demonstration path (green) when traditional TP-GMM/R is used and the relevant frames are identified and considered oriented by default. Similar to row (b), there is an improvement in destination reach for Tasks 1 and 2. However, the resultant path still fails to accomplish Task 4. Moreover, the path is inefficient in Task 2.

Row (d) shows the reproduced path (white) compared to the demonstration path (green) when ring TP-GMM/R is used and the relevant frames are identified to be orientation-less. The reproduced paths all reach their target locations and are smooth.

Table 4.4 The distances between the reproduced paths and the ground truth when all frames are considered OFs or when frames are identified as OFs or OLFs. The results are shown when all lead frames are used in TP-GMR or when only the relevant frames are used.

Task		d_{total}^{all}	d_{total}^{rel}	d_{start}^{all}	d_{start}^{rel}	d_{end}^{all}	d_{end}^{rel}
Task 1	All OFs	0.0474	0.0127	0.0135	0.000427	0.1012	0.0412
	OFs & OLFs	0.049	0.0113	0.005	0.0024	0.1172	0.0346
Task 2	All OFs	0.0737	0.0726	0.004	0.0039	0.1184	0.0086
	OFs & OLFs	0.0112	0.0097	0.0033	0.0078	0.0078	0.0058
Task 4	All OFs	0.0601	0.051	0.0139	0.0048	0.111	0.0747
	OFs & OLFs	0.0354	0.0248	0.0164	0.0177	0.0076	0.0121

Table 4.4 shows the distances between the demonstration paths and the reproduced path in multiple situations:

- d_{total}^{all} is the average distance between all the points on the reproduced path using all lead frames and the demonstration path. d_{total}^{rel} is the distance between all the points on the reproduced path using relevant frames and the demonstration path.
- d_{start}^{all} is the distance between the first point on the reproduced path using all lead frames and the demonstration path. d_{start}^{rel} is the distance the first point on the reproduced path using relevant frames and the demonstration path. The reason why this metric is calculated is because the start and end of the path often play a key role in the success of common tasks such as pick-and-place.
- d_{end}^{all} is the distance between the last point on the reproduced path using all lead frames and the demonstration path. d_{end}^{rel} is the distance the last point on the reproduced path using relevant frames and the demonstration path.

Each metric is calculated twice for each task:

- All OFs; when all the frames are considered to be oriented by default. That is, the normal Gaussian is used for modelling the paths.
- OFs and OLFs; when the frames are identified to be either OFs or OLFs. That is, the ring Gaussian is used to model the paths with respect to the OLFs.

In Table 4.4, the distance is highlighted:

- Green if the distance of the OFs and OLFs is less than that of the OFs alone. This means that our method has increased the similarity between the demonstration path and the reproduced path.
- Yellow if the distance of the OFs and OLFs is comparable to that of the OFs alone, i.e. with a difference of less than 0.01.
- Red if the distance of the OFs and OLFs is higher than that of the OFs alone.

The results show that using ring Gaussians generally improves or maintains the similarity between the demonstration path and the reproduced path compared to using the normal Gaussian alone.

4.4. Conclusion

The most common modelling used in TP-LfD is a GMM due to its mathematical properties and ability to model complex relations. However, a GMM fails to describe a path when it is equally occurring around a spatial task parameter, i.e. a frame. That is, a Gaussian will always be biased towards a certain side of a frame, and will not reflect that a path can occur on other sides.

In this chapter, we define a type of frames with respect to which paths are equally likely to occur in any direction. We name them orientation-less frames since if the frame changes orientation, the paths' functionality is not affected. Orientation-less frames are automatically identified based on their GMM. Their GMM does not include any Gaussian component that is narrow and stretched along the direction of the demonstration paths. That is, no portion of the path is constrained with respect to the frame's orientation.

To cater for orientation-less frames, we present a ring Gaussian modelling that gives equal probability for a path point around the frame. The ring Gaussian is a Gaussian that is revolved around a frame forming a ring. Ring Gaussians are calculated for orientation-less frames as part of the ring TP-GMM algorithm. To incorporate ring Gaussians with the normal Gaussian for oriented frames, we design an algorithm to convert a ring Gaussian to a normal Gaussian on a case-by-case basis during TP-GMR for path reproduction.

Using the presented ring Gaussian modelling, an improvement in paths is observed in tasks involving an orientation-less frame. The paths generated were smoother and more effective at reaching the target or source frames. Moreover, the algorithm was used to intuitively learn 3 different industrial tasks providing a reliable and flexible performance.

An orientation-less frame might be associated with an orientation-less real life object. Often, the frame might not be overlapping with the object's "centre", e.g. its grasping point. When modelling the paths with ring Gaussians that will be centred around the frame, the path might fail to reach the object's

“centre”. Therefore, in future works, we aim to adjust the modelling of the ring Gaussian such that it can be centred at a point offset from the frame, to be automatically identified. Moreover, we aim to improve the accuracy of the OLF identifier by employing a feedback system that adjusts decisions based on performance.

Chapter 5: Tool Integration and Extended Applications

5.1. Introduction

The work in chapter 3 and 4 was integrated together in a tool, GenLfD. GenLfD is a Generic tool that allows operators to intuitively use Learning from Demonstration to program cobots for industrial tasks. The tool is made to be used intuitively by operators. As part of this research, we have created a tutorial document to teach a user with minimal programming experience how to use the tool. The tutorial is in Appendix A. and the tool can be found online on: https://github.com/sherineza/GenLfD

As shown in chapters 3 and 4, the tool can be used for a range of industrial tasks involving 1 or 2 objects such as pick-and-place, gluing, and simple assembly operations. In this chapter, GenLfD is used to overcome potential issues: partial occlusion and obstacle avoidance. Moreover, in chapter 3 and 4, the experiments performed were in an industrial simulation environment. In this chapter, we evaluate our code in real-life settings.

5.2. Partial Occlusion

In chapter 3, visual frames are detected from 2D images of the task setting. Frames that belong to the same rigid object are grouped together as redundant frames. One of the advantages of grouping redundant frames together is that it solves the problem of partial occlusion in a new scenario. If a task-relevant object was partially occluded in the new image, its corresponding relevant frame might be undetectable. In that case, we resort to detecting the redundant frames belonging to the relevant frame's *object*. Redundant frames have a fixed relative position with respect to each other. Therefore, if one frame from an *object* is detected, the other redundant frames' positions can be estimated, including that of the relevant frame. However, if none of the frames from an *object* are found, the process is terminated.

Consider the task setting of Task 1 in chapter 3. The task is to pick the small circuit board (object 1) and place it in a fixed box on the side (not in camera view). Using the algorithm developed in this thesis, five demonstrations were recorded, visual features were detected, redundant features were grouped and finally, relevant features were identified (Figure). One of the relevant features identified belongs to object 1 while the other belongs to the background table, so by substitution, to the fixed box. The task is successfully learnt using TP-GMM and the path (in white) is reproduced using TP-GMR.

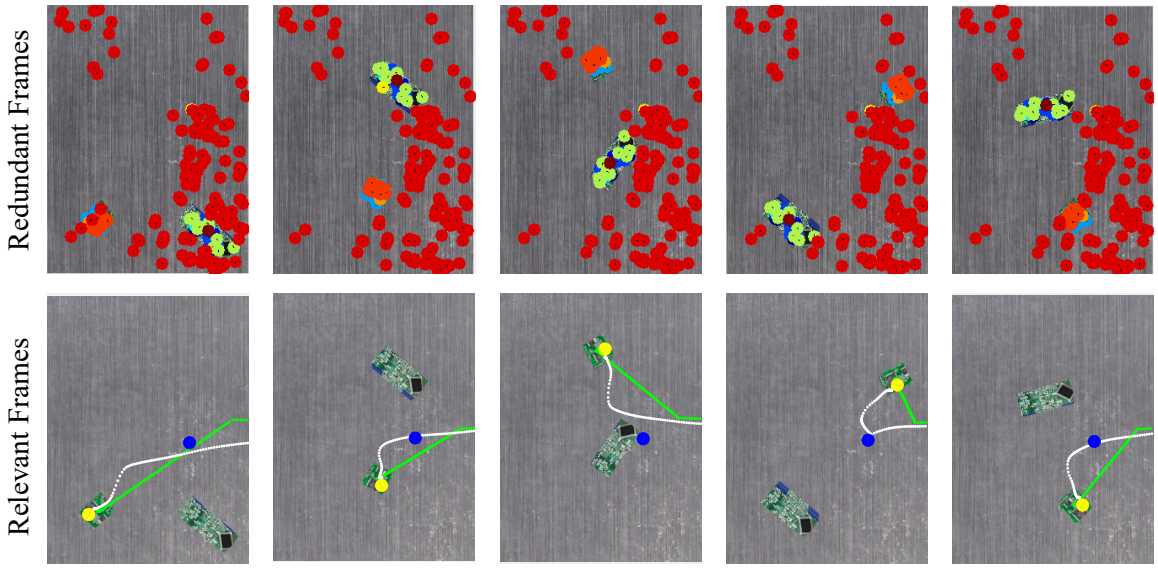


Figure 5.1 The recorded demonstrations of task 1.The first row of images shows the detected visual features grouped as redundant frames. The second row shows the identified relevant frames, an OLF belonging to the small circuit board and an OF belonging to the table. The demonstration path is shown in green and the reproduced path in white.

When reproducing the path in new settings, it is essential to identify the new positions of the relevant frames. Consider the case where one of the relevant frames, the one belonging to the table is occluded by the cobot's gripper. One of the redundant frames of the occluded frames is detected and the relative position between these two frames is used to calculate the hidden position of the occluded frame (Figure 5.2).

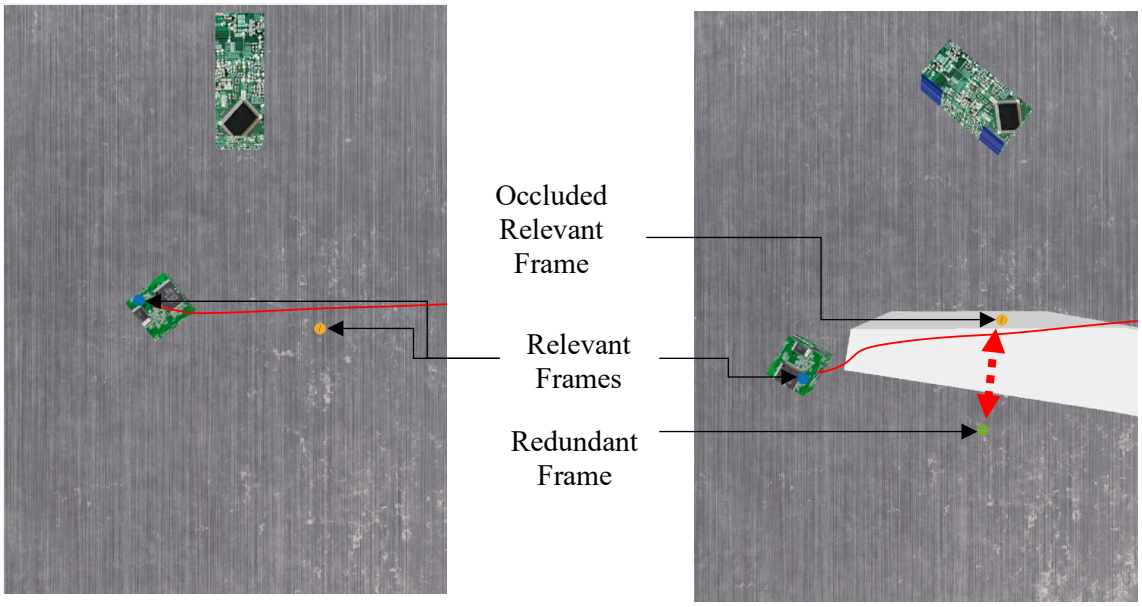


Figure 5.2 To overcome the occlusion of a relevant frame (the partial occlusion of an object), a visible redundant frame is detected and used to calculate the hidden position of the occluded frame.

5.3. Obstacle Avoidance

In chapter 4, ring Gaussians were formed to model paths with respect to orientation-less frames. If a frame belongs to an obstacle, the paths will maintain a distance from this frame. Hence, the calculated ring Gaussians will have a large diameter. When the path is reproduced, the obstacles large ring Gaussians will ensure that the reproduced path points tend towards maintaining a safe distance from the obstacle.

Consider the task setting where 2 circuit boards and a box are on a table in varying positions. The task is to pick the small circuit board (object 1) and place it in the box (object 2), while avoiding the second larger circuit board (obstacle). Five demonstrations are provided as shown in Figure 5.3. In some of the demonstrations, the obstacle is placed between object 1 and 2, whereas in other demonstrations, the obstacle is placed on the side. This is to teach the cobot that the demonstration path, shown in green, should pass around the obstacle only when it is in the way between object 1 and 2 and it is not really task related.

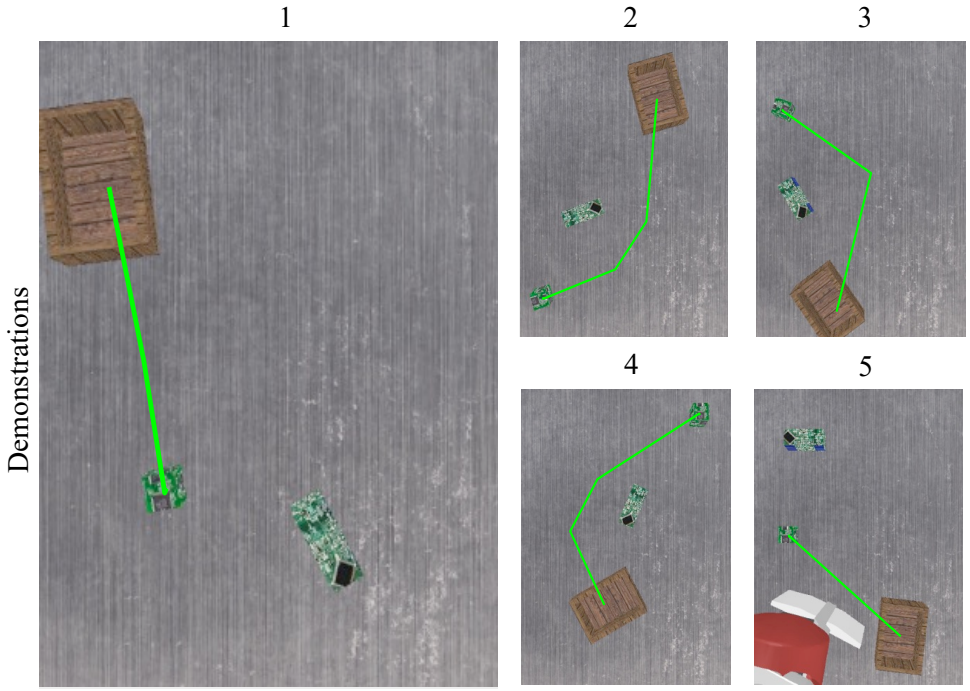


Figure 5.3 The demonstrations images and paths for the pick-and-place task with obstacle avoidance.

Visual features are automatically detected and matched in the demonstration images. Redundant features are groups together into objects with a lead frame from each object, as shown in Figure .

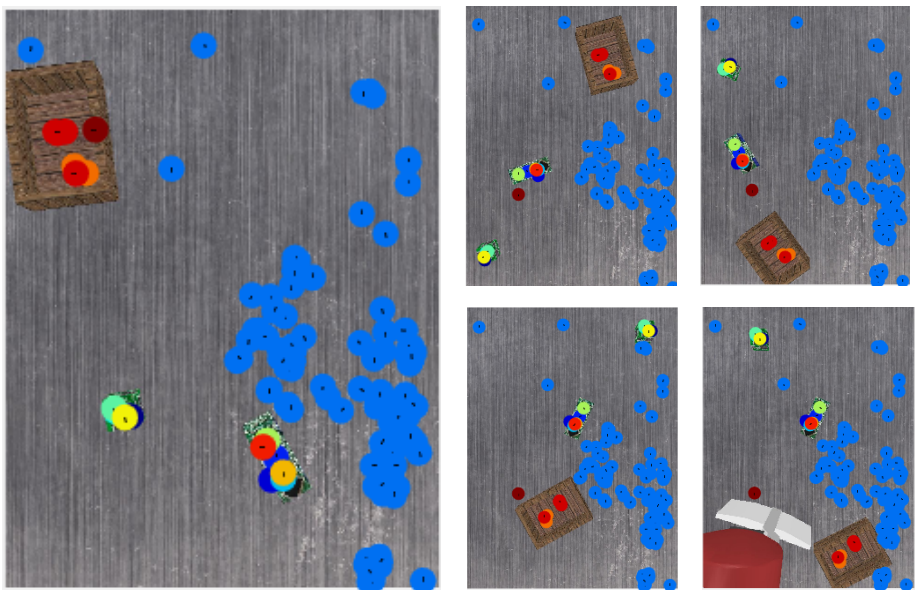


Figure 5.4 The detected frames from all the demonstration images. Redundant frames are matched into objects and given the same colour.

TP-GMM is used to calculate the GMM for each lead frame. The GMM is then used to identify if a frame is oriented or orientation-less, as shown in Figure 5.5 all the lead frames were identified as orientation-less. Ring Gaussians are calculated for the orientation-less frames. Figure 5.6 shown the ring Gaussians learnt for the 3 objects in the scene, object 1, object 2 and the obstacle. We notice that the smallest ring Gaussian around the obstacle is of large diameter, ensuring that the path remains far from the obstacle. The paths reproduced using the lead frames is shown in red in Figure 5.5. The paths are unsatisfactory; too curvy, do not reach the target and do not avoid the obstacle.

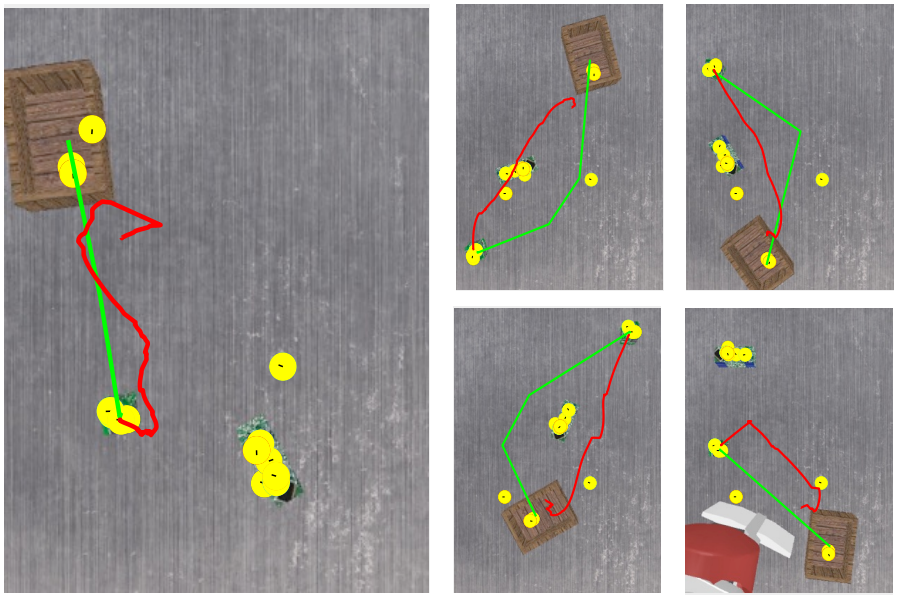


Figure 5.5 The reproduced paths, in red, using ring TP-GMM. All the lead frames are detected to be orientation-less frames.

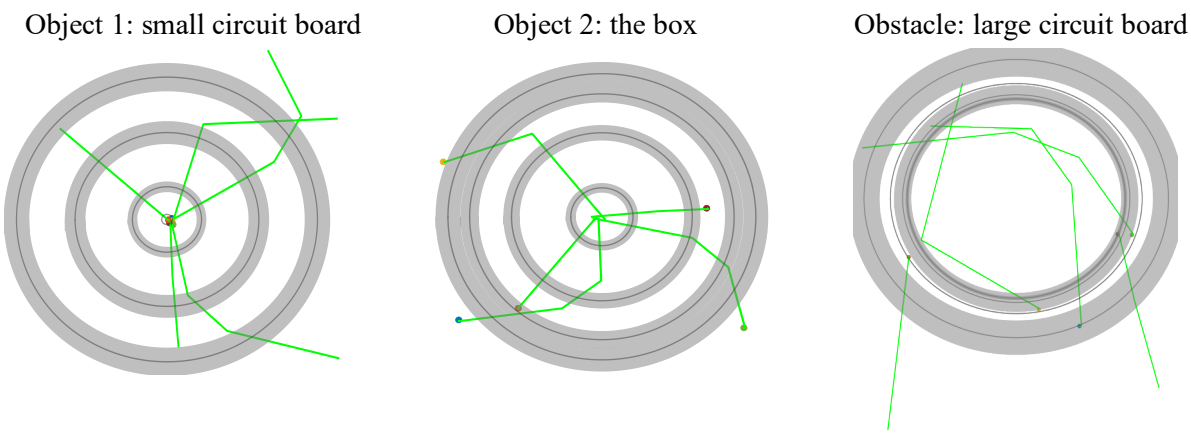


Figure 5.6 The calculated ring Gaussians for 3 of the lead frames belonging to each of the relevant objects: the small circuit board, the box and the large circuit board.

Assume that the user identifies the relevant frames to the task manually, including the known obstacle. This would have been done automatically using the reinforcement learning algorithm in chapter 3. However, that algorithm fails to detect obstacles as relevant frames due to the nature of the cost function. In future works, obstacles might need to be dynamically detected using depth sensors and avoided using the suggested algorithm. Figure 5.7 shows the reproduced paths (in white) when accounting for one frame each from object 1, object 2 and the obstacle. There is an improvement in the quality of the path compared to that in Figure 5.5. Most importantly, the obstacle is avoided on the path from object 1 to object 2.

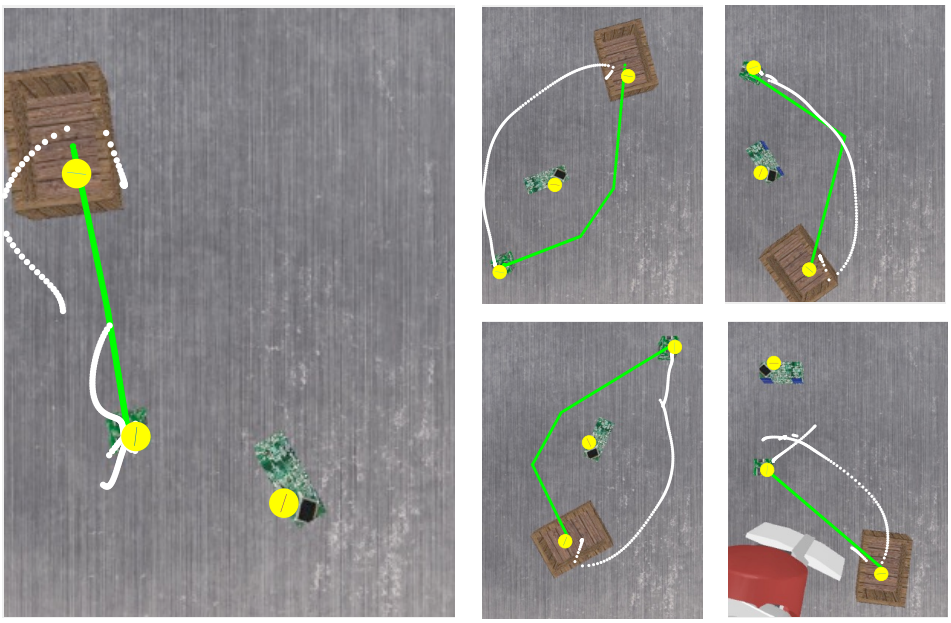


Figure 5.7 The reproduced paths using the relevant frames, one of which belongs to the known obstacle.

When reproducing the path, ring Gaussians have to be transformed into normal Gaussians before performing the weighted Gaussian product. That is done using the process described in chapter 4. However, in the case of an obstacle, the intersection between the obstacle's ring Gaussian and the other

objects' will be a Gaussian at a safe distance from the obstacle, such as in Figure 5.8. The "safe distance" is learnt from the demonstration since the Gaussians around the frame will have a large radius, hence a safe distance. There will be two points of intersection between the ring Gaussians, so an arbitrary one is chosen. However, it is important to make sure that the intersection point chosen on frame p's ring's intersection with frame j's is the same as the intersection between frame j's ring with frame p's. The resultant Gaussian combining these two intersections will also be at a safe distance from the obstacle. That way, the reproduced path will deviate from the obstacle.

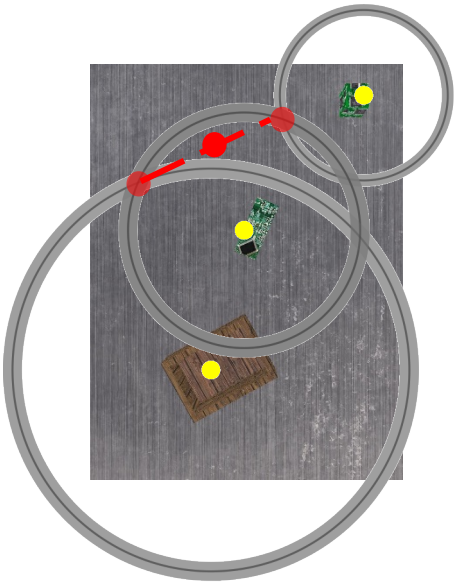


Figure 5.8 The ring Gaussians at time step t=101. The ring Gaussians are converted to the normal Gaussians by taking the intersection between every two pair of Gaussians. The resultant normal Gaussian therefore falls at a safe distance from the obstacle.

According to the results shown in Figure 5.7, the generated paths successfully avoided the obstacle in a smooth manner in demonstration 2, 3 and 4 when the obstacle was obstructing the path. However, when the obstacle was not obstructing the path, the reproduced path was functional in going from object 1 to object 2 but in a rough manner. Therefore, in future work, we aim to identify if an obstacle exists before accounting for its corresponding feature in the path reproduction algorithm.

5.4. Conclusion

In conclusion, the results presented in this chapter prove that the tool developed as part of this thesis, GenLfD, has the potential of overcoming typical robotic challenges: partial occlusion and obstacle avoidance.

To further improve the applicability of our tool with obstacle avoidance, a future advancement is to detect when an object is an obstacle. That way, two problems are avoided: 1) the deteriorating algorithm performance when an object is accounted for while reproducing a path when in reality it is not an

obstacle since it is out of the way; and 2) the ability to cater for multiple unpredictable obstacles rather than a predefined one.

Chapter 6: Conclusion

6.1. Thesis Motivation

In this research project, we aimed to bridge the gap between industrial deployment of collaborative robots (cobots) and their research application. It was noticed that research application focused on teaching cobots complex tasks involving human-awareness and uncertainties. Moreover, research also spanned the field of teaching cobots tasks intuitively using demonstrations or trial-and-error. However, industrial applications were limited to predictable tasks in proximity with a human. Moreover, industrial cobots were still programmed using their classical user-interface by specifying way points. Albeit intuitive, this programming technique only yields predictable motions by the cobot. Chapter 2 further elaborated on the different research technologies relating to cobots and some of the missing pieces preventing their implementation into industrial scenarios.

After the extensive literature review presented in **chapter 2**, a learning by demonstration algorithm known as task parametrized Gaussian mixture models/regression (TP-GMM/R) was chosen as a base for our work. TP-GMM/R is an intuitive method to program cobots to perform flexible tasks. It is capable of encoding path and time dependencies between objects or people in the scene. TP-GMM/R requires the end-user to simply provide demonstrations of a task being done to teach the task to the cobot. Therefore, it is dubbed as "intuitive". However, a back-end user will need to specify task parameters, which are positions of task relevant objects/locations in the scene. This usually requires a complex computer vision algorithm to be designed and customised for each task. This might hinder the use of TP-GMM/R to program cobots in industrial scenarios, especially in mass customisation, due to the time limitation that prevents the designing of a vision algorithm for each new task. Moreover, being given a wrong or sub-optimal choice of task parameters leads to a deterioration in learning performance.

6.2. Thesis Contributions

In **chapter 3**, we presented an algorithm that provides a generic task parameter detector and identifier. In brief, visual features, e.g. SURF features, were detected from demonstration images and matched between the images to be used as frames of reference. Frames belonging to the same object were grouped together to overcome partial occlusion and decrease the total number of features. Then, a reinforcement learning algorithm was used to identify the optimal frames to be used in TP-GMM/R to reproduce a path as close as possible to the demonstration paths.

The devised novel task parameter detection pipeline allowed us to program the cobot for multiple industrial tasks involving different objects without programmatically changing the algorithm. That is, TP-GMM/R was used without the need for a back-end user to custom design a computer vision algorithm to detect new objects for a new learnt task. Our work makes TP-GMM/R truly intuitive to

use, start to end, thus allowing it to be easily used on factory floors. This enables cobots to be used for more complex and flexible collaborative tasks in manufacturing.

Upon using TP-GMM/R to program tasks in chapter 3's experiments, we noticed a limitation imposed by the GMM modelling in TP-GMM/R. The GMM is composed of several Gaussians modelling the demonstration paths with respect to a frame of reference. If path points are scattered around a frame in all directions, then a Gaussian will not be able to capture all the points without being biased to a certain direction. Alternatively, if a Gaussian described all the points in all directions, then its centre will overlap with the frame's origin, thus falsely implying that the points are concentrated on the origin. Therefore, in **chapter 4**, we identified such a frame as "orientation-less" since its orientation is irrelevant to the path points. We designed a novel Gaussian model that can describe the points' variability around the frame and their distance from the frame. This novel Gaussian model, the ring Gaussian, is a Gaussian that is spanned around the frame. It accurately encodes the points' distance from the frame and the fact that they occur all around the frame.

The ring Gaussian was incorporated in the TP-GMM/R algorithm and used to program cobots for tasks such as pick-and-place, handover, sorting. The path reproduced using ring TP-GMM/R was smoother and more successful at accomplishing the task than the path reproduced using traditional TP-GMM/R. Therefore, the work in chapter 4 improved on the performance of TP-GMM/R for industrial tasks by improving the efficiency and effectiveness of the cobot paths generated. This makes TP-GMM/R more likely and susceptible to being used in programming cobots in flexible manufacturing settings.

Finally in **chapter 5,** the tool was used to overcome two common task challenges: partial occlusion and obstacle avoidance. We also provide a link to our code with tutorial steps in Appendix A on how to use it.

6.3. Safety Recommendations

Any implementation of human-robot collaboration must undergo risk assessment and comply with ISO 10218-1/2:2011 and ISO TS 15066. The former states four collaboration safety-operative modes: safety-rated monitored stop, hand guiding, speed and separation monitoring, and power and force limiting. Achieving these modes led to different fields of research such as collision avoidance (Lenz et al., 2009; Meziane et al., 2017; Wang et al., 2013; Schmidt and Wang, 2014), human motion prediction (Wang et al., 2017; Dinh et al., 2015), risk assessment through VR and simulation (Matsas et al., 2018) and other safety enabling technologies. The ISO 10218-1/2:2011 standards also necessitate performing risk assessment that is custom to each task, setup, and tools. ISO TS 15066 elaborates on the forces and pressures that are safe for different human body parts and how to calculate them. Together, these standards provide the methodology of performing risk assessment on collaborative robotic setups to make sure that the safety of the human operator is ensured.

In this thesis, the algorithms developed target the task programming of the robot without contributing to the safety factor. However, it does not contradict with any potential safety added feature. For example, the algorithms developed do not determine or limit joint velocities or torques, so a safety algorithm can freely control velocity and torque to ensure safety. It is a necessity that the implementation of the algorithms in this thesis are overlayed by a safety algorithm that is ready to override the robot's program in case of danger.

6.4. Future Prospects

The recommendations for future work include the following:

- Redundant frames are currently identified using an open-loop algorithm, by calculating the relative positions between every pair of frames. In the future, deep learning vision algorithms can be incorporated to validate the results of grouping redundant frames using intelligent object segmentation algorithms. A deep learning algorithm would provide more accurate, educated segmentation results that are robust to the object rotating about all axes and varying distance to the camera.
- Features detected are currently 2D which means that the path cannot be flexible with respect to their height change. Therefore, in the future, we aim to detect 3D features using a 3D camera and to include the gripper's orientation in the path points' data. That way the cobot will be able to learn more complex tasks involving variable orientation of objects and gripper orientation constraints.
- The ring Gaussian model currently implies that path points are occurring around a frame in all directions. However, in the future, we aim to design a partial ring Gaussian that spans a constrained portion of the orientation around the frame. This will more accurately describe real-life situations in which objects are not necessarily completely oriented or orientation-less, but rather something in between.
- Obstacle avoidance currently is only supported when the obstacle is predefined. In the future, an obstacle should be automatically detected so that no objects are considered in path production when they are not

Finally, this thesis has been a great journey, through which I have found myself in the field of learning from demonstration. I sincerely hope that the work done in this thesis will provide useful grounds for upcoming work on programming cobots for industrial tasks.

References

Abbeel, P., Coates, A., Quigley, M., & Ng, A. Y. (n.d.). *An application of reinforcement learning to aeroba.pdf*.

Admoni, H., Weng, T., Hayes, B., & Scassellati, B. (2016). Robot Nonverbal Behavior Improves Task Performance In Difficult Collaborations. *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*.51–58, doi: 10.1109/HRI.2016.7451733.

Agravante, D. J., Cherubini, A., Bussy, A., Gergondet, P., & Kheddar, A. (2014). Collaborative human-humanoid carrying using vision and haptic sensing. *Proceedings - IEEE International Conference on Robotics and Automation*, 607–612. https://doi.org/10.1109/ICRA.2014.6906917

Akgun, B., & Subramaman, K. (2011). *Robot Learning from Demonstration: Kinesthetic Teaching vs. Teleoperation*. 7. http://www.cc.gatech.edu/~ksubrama/files/HRIFinalBK.pdf

Alizadeh, T., & Karimi, N. (2018). Exploiting the task space redundancy in robot programming by demonstration. *Proceedings of 2018 IEEE International Conference on Mechatronics and Automation, ICMA 2018*, *August 2018*, 2394–2399. https://doi.org/10.1109/ICMA.2018.8484455

Alizadeh, T., & Malekzadeh, M. (2017). Identifying the relevant frames of reference in programming by demonstration using task-parameterized Gaussian mixture regression. *SII 2016 - 2016 IEEE/SICE International Symposium on System Integration*, 453–458. https://doi.org/10.1109/SII.2016.7844040

Alizadeh, T., & Saduanov, B. (2017). Robot programming by demonstration of multiple tasks within a common environment. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, *2017-Novem*(Mfi), 608–613. https://doi.org/10.1109/MFI.2017.8170389

Bänziger, T., Kunz, A., & Wegener, K. (2018). Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions. *Journal of Intelligent Manufacturing*, *31*(7), 1–14. https://doi.org/10.1007/s10845-018-1411-1

Baraglia, J., Cakmak, M., Nagai, Y., Rao, R., & Asada, M. (2016). Initiative in robot assistance during collaborative task execution. *ACM/IEEE International Conference on Human-Robot Interaction*, *2016-April*(March), 67–74. https://doi.org/10.1109/HRI.2016.7451735

Barattini, P., Morand, C., & Robertson, N. M. (2012). A proposed gesture set for the control of industrial collaborative robots. *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, 132–137. https://doi.org/10.1109/ROMAN.2012.6343743

Bay, H., Tuytelaars, T., & Gool, L. Van. (2006). SURF: Speeded Up Robust Features. *European*

Conference on Computer Vision 2006, 404–417.

Benzeghiba, M., De Mori, R., Deroo, O., Dupont, S., Erbes, T., Jouvet, D., Fissore, L., Laface, P., Mertins, A., Ris, C., Rose, R., Tyagi, V., & Wellekens, C. (2007). Automatic speech recognition and speech variability: A review. *Speech Communication*, *49*(10–11), 763–786. https://doi.org/10.1016/j.specom.2007.02.006

Bodden, C., Rakita, D., Mutlu, B., & Gleicher, M. (2016). Evaluating intent-expressive robot arm motion. *25th IEEE International Symposium on Robot and Human Interactive Communication, RO-MAN 2016*, *August*, 658–663. https://doi.org/10.1109/ROMAN.2016.7745188

Busch, B., Grizou, J., Lopes, M., & Stulp, F. (2017). Learning Legible Motion from Human–Robot Interactions. *International Journal of Social Robotics*, *9*(7).

Busch, B., Maeda, G., Mollard, Y., Demangeat, M., & Lopes, M. (2017). Postural optimization for an ergonomic human-robot interaction. *IEEE International Conference on Intelligent Robots and Systems*, *2017-Septe*, 2778–2785. https://doi.org/10.1109/IROS.2017.8206107

Calinon, S. (2016). A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, *9*(1), 1–29. https://doi.org/10.1007/s11370-015-0187-9

Calisgan, E., Haddadi, A., Van Der Loos, H. F. M., Alcazar, J. A., & Croft, E. A. (2012). Identifying nonverbal cues for automated human-robot turn-taking. *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, 418–423. https://doi.org/10.1109/ROMAN.2012.6343788

Cao, Zhe, Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*(Xxx), 1302–1310. https://doi.org/10.1109/CVPR.2017.143

Cao, Zhiqi, Hu, H., Zhao, Z., & Lou, Y. (2019). Robot programming by demonstration with local human correction for assembly. *IEEE International Conference on Robotics and Biomimetics, ROBIO 2019*, *December*, 166–171. https://doi.org/10.1109/ROBIO49542.2019.8961854

Carrus Home. (n.d.). *Nissan Motor Company: UR10 Cobots offer aging workfarce solution and reduce relief worker costs for global car manufacturer*. Retrieved November 20, 2020, from https://www.carrushome.com/en/nissan-motor-company-ur10-cobots-offer-aging-workforce-solution-and-reduce-relief-worker-costs-for-global-car-manufacturer/

Cesta, A., Orlandini, A., Bernardi, G., & Umbrico, A. (2016). Towards a planning-based framework for symbiotic human-robot collaboration. *IEEE International Conference on Emerging*

Technologies and Factory Automation, ETFA, *2016-Novem*. https://doi.org/10.1109/ETFA.2016.7733585

Chen, X., Zhang, X., Zhao, Z. Y., Yang, J. H., Lantz, V., & Wang, K. Q. (2007). Multiple hand gesture recognition based on surface EMG signal. *2007 1st International Conference on Bioinformatics and Biomedical Engineering, ICBBE*, 506–509. https://doi.org/10.1109/ICBBE.2007.133

Cheng, Y., Bao, J., Jia, Y., Deng, Z., Sun, Z., Bi, S., Li, C., & Xi, N. (2018). Modeling Natural Language Controlled Robotic Operations. *2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems, CYBER 2017*, 1072–1077. https://doi.org/10.1109/CYBER.2017.8446270

Coupeté, E., Manitsaris, F., & Manitsaris, S. (2019). Multi-users online recognition of technical gestures for natural Human-Robot Collaboration in manufacturing. *Autonomous Robots*, *43*(8).

de Gea Fernández, J., Mronga, D., Günther, M., Knobloch, T., Wirkus, M., Schröer, M., Trampler, M., Stiene, S., Kirchner, E., Bargsten, V., Bänziger, T., Teiwes, J., Krüger, T., & Kirchner, F. (2017). Multimodal sensor-based whole-body control for human–robot collaboration in industrial settings. *Robotics and Autonomous Systems*, *94*, 102–119. https://doi.org/10.1016/j.robot.2017.04.007

Devin, S., & Alami, R. (2016). An implemented theory of mind to improve human-robot shared plans execution. *ACM/IEEE International Conference on Human-Robot Interaction*, *2016-April*, 319–326. https://doi.org/10.1109/HRI.2016.7451768

Dīnēshchandra Jōshī, K., Chauhan, V., & Surgenor, B. (2020). A flexible machine vision system for small part inspection based on a hybrid SVM/ANN approach. *Journal of Intelligent Manufacturing*, *31*(1), 103–125.

Dinh, K. H., Oguz, O., Huber, G., Gabler, V., Wollherr, D. (2015). An approach to integrate human motion prediction into local obstacle avoidance in close human-robot collaboration. *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO), 1-6.*

Dumora, J., Geffard, F., Bidard, C., Aspragathos, N. A., & Fraisse, P. (2013). Robot assistance selection for large object manipulation with a human. *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, 1828–1833. https://doi.org/10.1109/SMC.2013.315

Duque, D. A., Prieto, F. A., & Hoyos, J. G. (2019). Trajectory generation for robotic assembly operations using learning by demonstration. *Robotics and Computer-Integrated Manufacturing*, *57*(July 2017), 292–302. https://doi.org/10.1016/j.rcim.2018.12.007

El Zaatari, S., Li, W., & Usman, Z. (2021). Ring Gaussian Mixture Modelling and Regression for Collaborative Robots. *Submitted to Robotics and Autonomous Systems*.

El Zaatari, S., Marei, M., Li, W., & Usman, Z. (2019). Cobot programming for collaborative industrial tasks: An overview. *Robotics and Autonomous Systems*, *116*, 162–180. https://doi.org/10.1016/j.robot.2019.03.003

El Zaatari, S., Wang, Y., Hu, Y., & Li, W. (2021). An Improved Approach of Task-Parameterized Learning from Demonstrations for Cobots in Dynamic Manufacturing. *Journal of Intelligent Manufacturing*.

El Zaatari, S., Wang, Y., Li, W., & Peng, Y. (2021). iTP-LfD: Improved Task Parametrised Learning from Demonstration for Generic Cobot Programming. *Robotics and Computer Integrated Manufacturing*, *69*.

Faber, M., Mertens, A., & Schlick, C. M. (2017). Cognition-enhanced assembly sequence planning for ergonomic and productive human–robot collaboration in self-optimizing assembly cells. *Production Engineering*, *11*(6).

Fischer, K., Kirstein, F., Jensen, L. C., Krüger, N., Kuklínski, K., Der Wieschen, M. V., & Savarimuthu, T. R. (2016). A comparison of types of robot control for programming by demonstration. *ACM/IEEE International Conference on Human-Robot Interaction*, *2016-April*, 213–220. https://doi.org/10.1109/HRI.2016.7451754

Forbes, M., Chung, M. J., Cakmak, M., Rao, R. P. N., & Way, N. E. S. (2014). Robot Programming by Demonstration with Crowdsourced Action Fixes. *HCOMP*.

Gabler, V., Stahl, T., Huber, G., Oguz, O., & Wollherr, D. (2017). A game-theoretic approach for adaptive action selection in close proximity human-robot-collaboration. *Proceedings - IEEE International Conference on Robotics and Automation*, 2897–2903. https://doi.org/10.1109/ICRA.2017.7989336

Gaz, C., Magrini, E., & De Luca, A. (2018). A model-based residual approach for human-robot collaboration during manual polishing operations. *Mechatronics*, *55*(February), 234–247. https://doi.org/10.1016/j.mechatronics.2018.02.014

Ghalamzan E., A. M., & Ragaglia, M. (2018). Robot learning from demonstrations: Emulation learning in environments with moving obstacles. *Robotics and Autonomous Systems*, *101*(February 2018), 45–56. https://doi.org/10.1016/j.robot.2017.12.001

Giuliani, M., & Knoll, A. (2013). Using embodied multimodal fusion to perform supportive and instructive robot roles in human-robot interaction. *International Journal of Social Robotics*, *5*(3),

345– 356.

Gleeson, B., Maclean, K., Croft, E., & Alcazar, J. (2013). Gestures for Industry: Intuitive human-robot communication from human observation. *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI).* 349–356.

Gombolay, M., Bair, A., Huang, C., & Shah, J. (2017). Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences. *International Journal of Robotics Research*, *36*(5–7), 597–617. https://doi.org/10.1177/0278364916688255

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org

Gu, S., Holly, E., Lillicrap, T., & Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *Proceedings - IEEE International Conference on Robotics and Automation*, 3389–3396. https://doi.org/10.1109/ICRA.2017.7989385

Gu, Y., Sheng, W., Crick, C., & Ou, Y. (2018). Automated assembly skill acquisition and implementation through human demonstration. *Robotics and Autonomous Systems*, *99*, 1–16. https://doi.org/10.1016/j.robot.2017.10.002

Gu, Y., Thobbi, A., & Sheng, W. (2011). Human-robot Collaborative Manipulation through Imitation and Reinforcement Leaming. *2011 IEEE International Conference on Information and Automation.*

Guerin, K. R., Riedel, S. D., Bohren, J., & Hager, G. D. (2014). Adjutant: A framework for flexible human-machine collaborative systems. *IEEE International Conference on Intelligent Robots and Systems*, *Iros*, 1392–1399. https://doi.org/10.1109/IROS.2014.6942739

Gustavsson, P., Syberfeldt, A., Brewster, R., & Wang, L. (2017). Human-robot Collaboration Demonstrator Combining Speech Recognition and Haptic Control. *Procedia CIRP*, *63*, 396–401.

Hamabe, T., Goto, H., & Miura, J. (2015). A programming by demonstration system for human-robot collaborative assembly tasks. *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, 1195–1201. https://doi.org/10.1109/ROBIO.2015.7418934

Hangl, S., Dunjko, V., Briegel, H. J., & Piater, J. (2017). *Skill Learning by Autonomous Robotic Playing using Active Learning and Creativity*. 1–15.

Hangl, S., Mennel, A., & Piater, J. (2017). *A novel Skill-based Programming Paradigm based on Autonomous Playing and Skill-centric Testing*.

Hawkins, K. P., Vo, N., Bansal, S., & Bobick, A. F. (2015). Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration. *IEEE-RAS International Conference on Humanoid Robots*, *2015-Febru*(February), 499–506. https://doi.org/10.1109/HUMANOIDS.2013.7030020

Hewitt, A., Yang, C., Li, Y., Cui, R. (2017). DMP and GMR based teaching by demonstration for a KUKA LBR robot. *Proceedings of the 2017 IEEE International Conference on Automation & Computing.*

Hu, B., & Chen, J. (2017). Optimal Task Allocation for Human-Machine Collaborative Manufacturing Systems. *IEEE Robotics and Automation Letters*, *2*(4), 1933–1940. https://doi.org/10.1109/LRA.2017.2714981

Hu, S., & Kuchenbecker, K. J. (2019). Hierarchical task-parameterized learning from demonstration for collaborative object movement. *Applied Bionics and Biomechanics*, *2019*. https://doi.org/10.1155/2019/9765383

Huang, C. M., Cakmak, M., & Mutlu, B. (2015). Adaptive coordination strategies for human-robot handovers. *Robotics: Science and Systems*, *11*(August). https://doi.org/10.15607/RSS.2015.XI.031

Huang, C. M., & Mutlu, B. (2016). Anticipatory robot control for efficient human-robot collaboration. *ACM/IEEE International Conference on Human-Robot Interaction*, *2016-April*(Section V), 83–90. https://doi.org/10.1109/HRI.2016.7451737

Huang, Y., Abu-Dakka, F. J., Silvério, J., & Caldwell, D. G. (2019). Generalized orientation learning in robot task space. *Proceedings - IEEE International Conference on Robotics and Automation*, *2019-May*(February), 2531–2537. https://doi.org/10.1109/ICRA.2019.8793540

Huang, Y., Silverio, J., Rozo, L., & Caldwell, D. G. (2018). Generalized task-parameterized skill learning. *Proceedings - IEEE International Conference on Robotics and Automation*, *May*, 5667–5674. https://doi.org/10.1109/ICRA.2018.8461079

Jia, Z., Lin, M., Chen, Z., & Jian, S. (2020). Vision-based robot manipulation learning via human demonstrations. *ArXiv*.

Johannsmeier, L., & Haddadin, S. (2017). A Hierarchical Human-Robot Interaction-Planning Framework for Task Allocation in Collaborative Industrial Assembly Processes. *IEEE Robotics and Automation Letters*, *2*(1), 41–48. https://doi.org/10.1109/LRA.2016.2535907

Kim, W., Lee, J., Peternel, L., Tsagarakis, N., & Ajoudani, A. (2018). Anticipatory Robot Assistance for the Prevention of Human Static Joint Overloading in Human-Robot Collaboration. *IEEE*

Robotics and Automation Letters, *3*(1), 68–75. https://doi.org/10.1109/LRA.2017.2729666

Kobayashi, H., Yasuda, T., Igarashi, H., & Suzuki, S. (2020). Language Use in Joint Action : The Means of Referring Expressions. *International Journal of Social Robotics*, *12*(5), 1021–1029. https://doi.org/10.1007/s12369-017-0462-3

Koch, P. J., van Amstel, M. K., Dębska, P., Thormann, M. A., Tetzlaff, A. J., Bøgh, S., & Chrysostomou, D. (2017). A Skill-based Robot Co-worker for Industrial Maintenance Tasks. *Procedia Manufacturing*, *11*(June), 83–90. https://doi.org/10.1016/j.promfg.2017.07.141

Kouris, A., Dimeas, F., & Aspragathos, N. (2018). A Frequency Domain Approach for Contact Type Distinction in Human-Robot Collaboration. *IEEE Robotics and Automation Letters*, *3*(2), 720–727. https://doi.org/10.1109/LRA.2017.2789249

Kramberger, A., Iturrate, I., Denisa, M., Mathiesen, S., & Sloth, C. (2020). Adapting Learning by Demonstration for Robot Based Part Feeding Applications. *Proceedings of the 2020 IEEE/SICE International Symposium on System Integration*, 954–959.

KUKA. (n.d.). *Innovative Škoda factory succeeds with KUKA LBR iiwa*. Retrieved November 20, 2020, from https://www.kuka.com/en-de/press/news/2017/02/innovative-skoda-factory-succeeds-with-kuka-lbr-iiwa

Kumičáková, D., Rengevič, A., Císar, M., & Tlach, V. (2017). Utilisation of Kinect Sensors for the Design of a Human-Robot Collaborative Workcell. *Advances in Science and Technology Research Journal*, *11*(4), 270–278. https://doi.org/10.12913/22998624/80937

Lafleche, J. F., Saunderson, S., & Nejat, G. (2019). Robot Cooperative Behavior Learning Using Single-Shot Learning From Demonstration and Parallel Hidden Markov Models. *IEEE Robotics and Automation Letters*, *4*(2), 193–200. https://doi.org/10.1109/LRA.2018.2885584

Lenz, C., Rickert, M., Panin, G., & Knoll, A. (2009). Constraint task-based control in industrial settings. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 3058-3063.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., & Quillen, D. (2018). Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International Journal of Robotics Research*, *37*(4–5), 421–436. https://doi.org/10.1177/0278364917710318

Li, Y., & Ge, S. S. (2014). Human-robot collaboration based on motion intention estimation. *IEEE/ASME Transactions on Mechatronics*, *19*(3), 1007–1014. https://doi.org/10.1109/TMECH.2013.2264533

Li, Y., Tee, K. P., Chan, W. L., Yan, R., Chua, Y., & Limbu, D. K. (2015). Role adaptation of human

and robot in collaborative tasks. *Proceedings - IEEE International Conference on Robotics and Automation*, *2015-June*(June), 5602–5607. https://doi.org/10.1109/ICRA.2015.7139983

Liang, Y. S., Pellier, D., Fiorino, H., & Pesty, S. (2017). Evaluation of a robot programming framework for non-experts using symbolic planning representations. *RO-MAN 2017 - 26th IEEE International Symposium on Robot and Human Interactive Communication*, *2017-Janua*, 1121–1126. https://doi.org/10.1109/ROMAN.2017.8172444

Lichiardopol, S., Van De Wouw, N., & Nijmeijer, H. (2009). Control scheme for human-robot co-manipulation of uncertain, time-varying loads. *Proceedings of the American Control Conference*, *May*, 1485–1490. https://doi.org/10.1109/ACC.2009.5160062

Liu, H., & Wang, L. (2018). Gesture recognition for human-robot collaboration: A review. *International Journal of Industrial Ergonomics*, *68*, 355–367.

Luo, R., Hayne, R., & Berenson, D. (2018). Unsupervised early prediction of human reaching for human–robot collaboration in shared workspaces. *Autonomous Robots*, *42*, 631–648.

Maeda, G., Ewerton, M., Neumann, G., Lioutikov, R., & Peters, J. (2017). Phase estimation for fast action recognition and trajectory generation in human–robot collaboration. *The International Journal of Robotics Research*, *36*(13–14).

Maeda, G., Maloo, A., Ewerton, M., Lioutikov, R., & Peters, J. (2016). Anticipative Interaction Primitives for Human-Robot Collaboration. *The 2016 AAAI Fall Symposium Series*.

Magrini, E., Flacco, F., & De Luca, A. (2015). Control of generalized contact motion and force in physical human-robot interaction. *Proceedings - IEEE International Conference on Robotics and Automation*, *2015-June*(June), 2298–2304. https://doi.org/10.1109/ICRA.2015.7139504

Makrini, I. El, Merckaert, K., Lefeber, D., & Vanderborght, B. (2017). Design of a collaborative architecture for human-robot assembly tasks. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Marvel, J. A., Bagchi, S., Zimmerman, M., & Antonishek, B. (2020). Towards effective interface designs for collaborative HRI in manufacturing. *ACM Transactions on Human-Robot Interaction*, *9*(4). https://doi.org/10.1145/3385009

Masood, T., & Sonntag, P. (2020). Industry 4.0: Adoption challenges and benefits for SMEs. *Computers in Industry*, *121*, 103261. https://doi.org/10.1016/j.compind.2020.103261

Matsas, E., Vosniakos, G. C., Batras, D. (2018). Prototyping proactive and adaptive techniques for human-robot collaboration in manufacturing using virtual reality. *Robotics and Computer-Integrated Manufacturing,* 50, 168-180.

Maurice, P., Huber, M. E., Hogan, N., & Sternad, D. (2018). Velocity-Curvature Patterns Limit Human-Robot Physical Interaction. *IEEE Robotics and Automation Letters*, *3*(1), 249–256. https://doi.org/10.1109/LRA.2017.2737048

Maurtua, I., Fernández, I., Tellaeche, A., Kildal, J., Susperregi, L., Ibarguren, A., & Sierra, B. (2017). Natural multimodal communication for human-robot collaboration. *International Journal of Advanced Robotic Systems*, *14*(4), 1–12. https://doi.org/10.1177/1729881417716043

Meziane, R., Otis, M. J. D., Ezzaidi, H. (2017). Human-robot collaboration while sharing production activities in dynamic environment: SPADER system. *Robotics and Computer-Integrated Manufacturing,* 48, 243-253.

Milliez, G., Lallement, R., Fiore, M., & Alami, R. (2016). Using human knowledge awareness to adapt collaborative plan generation, explanation and monitoring. *ACM/IEEE International Conference on Human-Robot Interaction*, *2016-April*, 43–50. https://doi.org/10.1109/HRI.2016.7451732

Mohan, V., & Bhat, A. A. (2018). Joint goal human robot collaboration-from remembering to inferring. *Procedia Computer Science*, *123*, 579–584. https://doi.org/10.1016/j.procs.2018.01.089

Muja, M., & Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and Applications*, *1*, 331–340. https://doi.org/10.5220/0001787803310340

Müller, R., Vette, M., & Mailahn, O. (2016). Process-oriented Task Assignment for Assembly Processes with Human-robot Interaction. *Procedia CIRP*, *44*, 210–215. https://doi.org/10.1016/j.procir.2016.02.080

Nakata, S., Kobayashi, H., Yasuda, T., Kumata, M., Suzuki, S., & Igarashi, H. (2011). Relation between skill acquisition and task specific human speech in collaborative work. 2011 RO-MAN. 337–342.

Nakata, S., Kobayashi, H., Kumata, M., & Suzuki, S. (2011) Human speed ontology changes in virtual collaborative work. *2011 4th International Conference on Human System Interactions (HSI),* 363-368.

Neto, P., Simão, M., Mendes, N., & Safeea, M. (2019). Gesture-based human-robot interaction for human assistance in manufacturing. *The International Journal of Advanced Manufacturing Technology*, *101*, 119–135.

Nikolaidis, S., Lasota, P., Ramakrishnan, R., & Shah, J. (2015). Improved human–robot team

performance through cross-training, an approach inspired by human team training practices. *The International Journal of Robotics Research*, *34*(14), 1711–1730.

Noohi, E., Zefran, M., & Patton, J. L. (2016). A Model for Human-Human Collaborative Object Manipulation and Its Application to Human-Robot Interaction. *IEEE Transactions on Robotics*, *32*(4), 880–896. https://doi.org/10.1109/TRO.2016.2572698

Nuzzi, C., Pasinetti, S., Lancini, M., Docchio, F., & Sansoni, G. (2019). Deep learning-based hand gesture recognition for collaborative robots. *IEEE Instrumentation & Measurement Magazine*, *22*, 44–51.

Park, K. B., Kim, M., Choi, S. H., & Lee, J. Y. (2020). Deep learning-based smart task assistance in wearable augmented reality. *Robotics and Computer-Integrated Manufacturing*, *63*(November 2019), 101887. https://doi.org/10.1016/j.rcim.2019.101887

Paxton, C., Hundt, A., Jonathan, F., Guerin, K., & Hager, G. D. (2017). CoSTAR: Instructing collaborative robots with behavior trees and vision. *Proceedings - IEEE International Conference on Robotics and Automation*, 564–571. https://doi.org/10.1109/ICRA.2017.7989070

Pedersen, M. R., Herzog, D. L., & Krüger, V. (2014). Intuitive skill-level programming of industrial handling tasks on a mobile manipulator. *IEEE International Conference on Intelligent Robots and Systems*, *Iros*, 4523–4530. https://doi.org/10.1109/IROS.2014.6943203

Pedersen, M. R., Nalpantidis, L., Andersen, R. S., Schou, C., Bøgh, S., Krüger, V., & Madsen, O. (2016). Robot skills for manufacturing: From concept to industrial deployment. *Robotics and Computer-Integrated Manufacturing*, *37*, 282–291. https://doi.org/10.1016/j.rcim.2015.04.002

Pellegrinelli, S., Moro, F. L., Pedrocchi, N., Molinari Tosatti, L., & Tolio, T. (2016). A probabilistic approach to workspace sharing for human–robot cooperation in assembly tasks. *CIRP Annals - Manufacturing Technology*, *65*(1), 57–60. https://doi.org/10.1016/j.cirp.2016.04.035

Penumuru, D. P., Muthuswamy, S., & Karumbu, P. (2020). Identification and classification of materials using machine vision and machine learning in the context of industry 4.0. *Journal of Intelligent Manufacturing*, *31*, 1229–1241.

Perez-D'Arpino, C., & Shah, J. A. (2017). C-LEARN: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. *Proceedings - IEEE International Conference on Robotics and Automation*, *1*(c), 4058–4065. https://doi.org/10.1109/ICRA.2017.7989466

Peternel, L., Tsagarakis, N., Caldwell, D., & Ajoudani, A. (2016). Adaptation of robot physical behaviour to human fatigue in human-robot co-manipulation. *IEEE-RAS International*

Conference on Humanoid Robots, 489–494. https://doi.org/10.1109/HUMANOIDS.2016.7803320

Peternel, L., Kim, W., Babic, J., & Ajoudani, A. (2017). Towards Ergonomic Control of Human-Robot Co-Manipulation and Handover. *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids),.*

Pohlt, C., Hell, S., Schlegl, T., & Wachsmuth, S. (2017). Impact of Spontaneous Human Inputs during Gesture based Interaction on a Real-World Manufacturing Scenario. *HAI 2017*.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., & Levine, S. (2018). *Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations*.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *ArXiv*.

Reeco. (n.d.). *BMW – Human Robot Collaboration*. Retrieved November 20, 2020, from https://www.reeco.co.uk/projects/bmw-mini-robotic-riveting/

Reyes, M. E., Meza, I. V., & Pineda, L. A. (2019). Robotics facial expression of anger in collaborative human–robot interaction. *International Journal of Advanced Robotic Systems*, *16*(1), 1–13. https://doi.org/10.1177/1729881418817972

ROBOTIQ. (2020). *Collaborative Robot Buyer's Guide*. *May*, 53. https://blog.robotiq.com/collaborative-robot-ebook

Rogowski, A., & Skrobek, P. (2020). Object identification for task-oriented communication with industrial robots. *Sensors (Switzerland)*, *20*(6). https://doi.org/10.3390/s20061773

Rossano, G. F., Martinez, C., Hedelind, M., Murphy, S., & Fuhlbrigge, T. A. (2013). Easy robot path programming concepts: An industrial perspective on path creation. *2013 44th International Symposium on Robotics, ISR 2013*, 1119–1126. https://doi.org/10.1109/ISR.2013.6695710

Rozo, L., Calinon, S., Caldwell, D. G., Jiménez, P., & Torras, C. (2016). Learning Physical Collaborative Robot Behaviors From Human Demonstrations. *IEEE Transactions on Robotics*, *32*(3), 513–527. https://doi.org/10.1109/TRO.2016.2540623

Sadrfaridpour, B., & Wang, Y. (2018). Collaborative Assembly in Hybrid Manufacturing Cells: An Integrated Framework for Human-Robot Interaction. *IEEE Transactions on Automation Science and Engineering*, *15*(3), 1178–1192. https://doi.org/10.1109/TASE.2017.2748386

Schmidbauer, C., Komenda, T., & Schlund, S. (2020). Teaching cobots in learning factories - User and usability-driven implications. *Procedia Manufacturing*, *45*, 398–404. https://doi.org/10.1016/j.promfg.2020.04.043

Schmidt. B., Wang. L. (2014) Depth camera based collision avoidance via active robot control. *Journal of Manufacturing Systems,* 33(4), 711-718.

Schou, C., Damgaard, J. S., Bogh, S., & Madsen, O. (2013). Human-robot interface for instructing industrial tasks using kinesthetic teaching. *2013 44th International Symposium on Robotics, ISR 2013*. https://doi.org/10.1109/ISR.2013.6695599

Schou, Casper, Andersen, R. S., Chrysostomou, D., Bøgh, S., & Madsen, O. (2018). Skill-based instruction of collaborative robots in industrial settings. *Robotics and Computer-Integrated Manufacturing*, *53*(June 2016), 72–80. https://doi.org/10.1016/j.rcim.2018.03.008

Schulz, R., Kratzer, P., & Toussaint, M. (2017). Building a bridge with a robot: A system for collaborative on-table task execution. *HAI 2017 - Proceedings of the 5th International Conference on Human Agent Interaction*, 399–403. https://doi.org/10.1145/3125739.3132606

Schwarz, G. E. (1978). Estimating the dimension of a model. *The Annals of Statistics,* 6(2).

Sena, A., Michael, B., & Howard, M. (2019). Improving task-parameterised movement learning generalisation with frame-weighted trajectory generation. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

She, L., & Chai, J. Y. (2017). Interactive learning of grounded verb semantics towards human-robot communication. *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, *1*, 1634–1644. https://doi.org/10.18653/v1/P17-1150

Sheng, W., Thobbi, A., & Gu, Y. (2015). An Integrated Framework for Human-Robot Collaborative Manipulation. *IEEE Transactions on Cybernetics*, *45*(10), 2030–2041. https://doi.org/10.1109/TCYB.2014.2363664

Shukla, D., Erkent, O., & Piater, J. (2017). Proactive, incremental learning of gesture-Action associations for human-robot collaboration. *RO-MAN 2017 - 26th IEEE International Symposium on Robot and Human Interactive Communication*, *2017-Janua*, 346–353. https://doi.org/10.1109/ROMAN.2017.8172325

Silverio, J., Calinon, S., Rozo, L., & Caldwell, D. G. (2019). Learning Task Priorities from Demonstrations. *IEEE Transactions on Robotics*, *35*(1), 78–94. https://doi.org/10.1109/TRO.2018.2878355

Srimal, A. S., Muthugala, V. J., & Jayasekara, B. P. (2017). *Deictic Gesture Enhanced Fuzzy Spatial Relation Grounding in Natural Language*.

Steinmetz, F., & Weitschat, R. (2016). Skill parametrization approaches and skill architecture for

human-robot interaction. *IEEE International Conference on Automation Science and Engineering*, *2016-Novem*, 280–285. https://doi.org/10.1109/COASE.2016.7743419

Steinmetz, F., Wollschlager, A., & Weitschat, R. (2018). RAZER-A HRI for Visual Task-Level Programming and Intuitive Skill Parameterization. *IEEE Robotics and Automation Letters*, *3*(3), 1362–1369. https://doi.org/10.1109/LRA.2018.2798300

Tang, G., Webb, P., & Thrower, J. (2019). The development and evaluation of Robot Light Skin: A novel robot signalling system to improve communication in industrial human–robot collaboration. *Robotics and Computer-Integrated Manufacturing*, *56*(February 2018), 85–94. https://doi.org/10.1016/j.rcim.2018.08.005

Tareen, S. A. K., & Saleem, Z. (2018). A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. *2018 International Conference on Computing, Mathematics and Engineering Technologies: Invent, Innovate and Integrate for Socioeconomic Development, ICoMET 2018 - Proceedings*, *2018-Janua*, 1–10. https://doi.org/10.1109/ICOMET.2018.8346440

Unhelkar, V. V, Yang, X. J., & Shah, J. (2017). Challenges for Communication Decision-Making in Sequential Human-Robot Collaborative Tasks. *RSS 2017 Workshop on Mathematical Models, Algorithms, and Human-Robot Interaction*, 28–31.

Vidaković, J., Jerbić, B., Šekoranja, B., Švaco, M., & Šuligoj, F. (2020). Learning from Demonstration Based on a Classification of Task Parameters and Trajectory Optimization. *Journal of Intelligent & Robotic Systems Volume*, *99*, 261–275.

Vogt, D., Stepputtis, S., Grehl, S., Jung, B., & Ben Amor, H. (2017). A system for learning continuous human-robot interactions from human-human demonstrations. *Proceedings - IEEE International Conference on Robotics and Automation*, *May*, 2882–2889. https://doi.org/10.1109/ICRA.2017.7989334

Wang, L., Schmidt, B., & Nee, A. Y. C. (2013). Vision-guided active collision avoidance for human-robot collaborations. *Manufacturing Letters,* 1(1), 5-8.

Wang, Y., Ye, X., Yang, Y., Zhang, W. (2017). Collision-free trajectory planning in human-robot interaction hrough hand movement prediction from vision. *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids),* 305-310.

Willibald, C. (2020). *Development of an interactive robot programming method*. Technische Universität München.

Wojtara, T., Uchihara, M., Murayama, H., Shimoda, S., Sakai, S., Fujimoto, H., & Kimura, H. (2009). Human-robot collaboration in precise positioning of a three-dimensional object. *Automatica*,

45(2), 333–342. https://doi.org/10.1016/j.automatica.2008.08.021

Wongphati, M., Osawa, H., & Imai, M. (2015). Gestures for manually controlling a helping hand robot. *International Journal of Social Robotics*, *7*(5), 731–742.

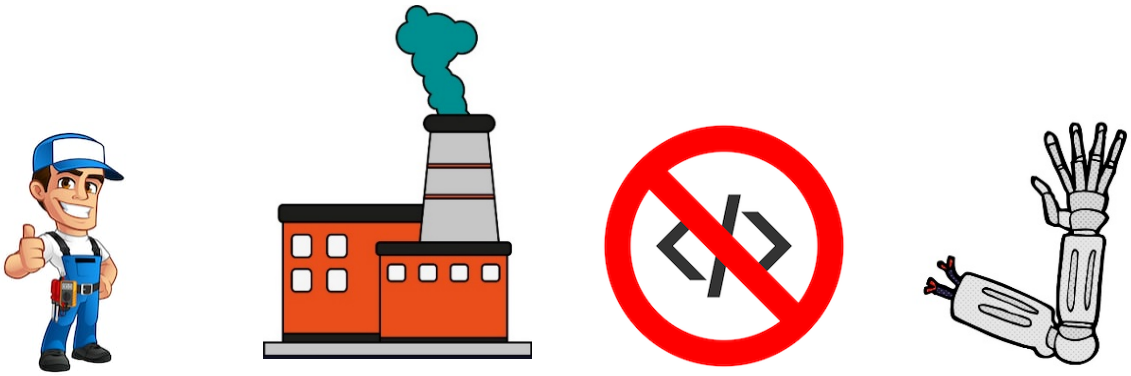
Yang, C., Zeng, C., Liang, P., Li, Z., Li, R., & Su, C.-Y. (2017). Interface Design of a Physical Human-Robot Interaction System for Human Impedance Adaptive Skill Transfer. *IEEE Transactions on Automation Science and Engineering*, *99*, 1–12.

Yu, T., Huang, J., & Chang, Q. (2020) Mastering the working sequence in human-robot collaborative assembly based on reinforcement learning.

Zhu, H., Gabler, V., & Wollherr, D. (2017). Legible action selection in human-robot collaboration. *RO-MAN 2017 - 26th IEEE International Symposium on Robot and Human Interactive Communication*, *2017-Janua*(August), 354–359. https://doi.org/10.1109/ROMAN.2017.8172326

Appendix A: Instructions

This appendix include the tutorial document attached with the integrated tool to teach a user how to program a cobot using the tool.



You are a technician in a factory. You are required to program a robot. You have no programming experience.

You are presented with a tool called **GenLfD**, which allows you to program robots for Generic tasks using Learning from Demonstrations.

To be able to run the program, you need to have installed:

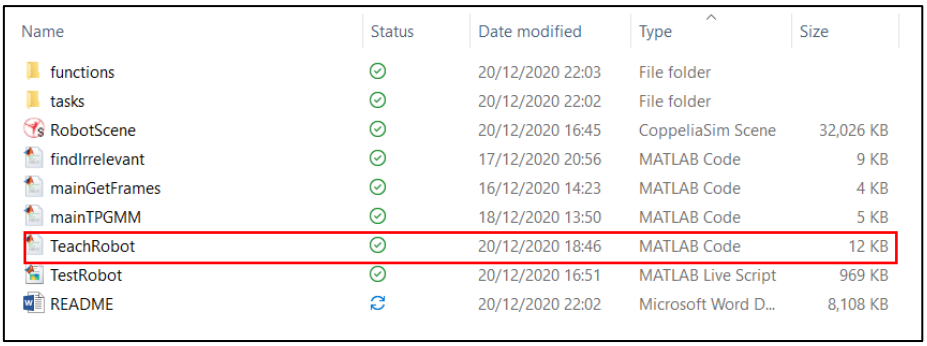
1. MATLAB R2020a
2. CoppeliaSim EDU (free download https://www.coppeliarobotics.com/downloads)

Step 1

The tutorial folder includes different files and folder including code functions, simulation files and saved examples. As a user, you will only be using:

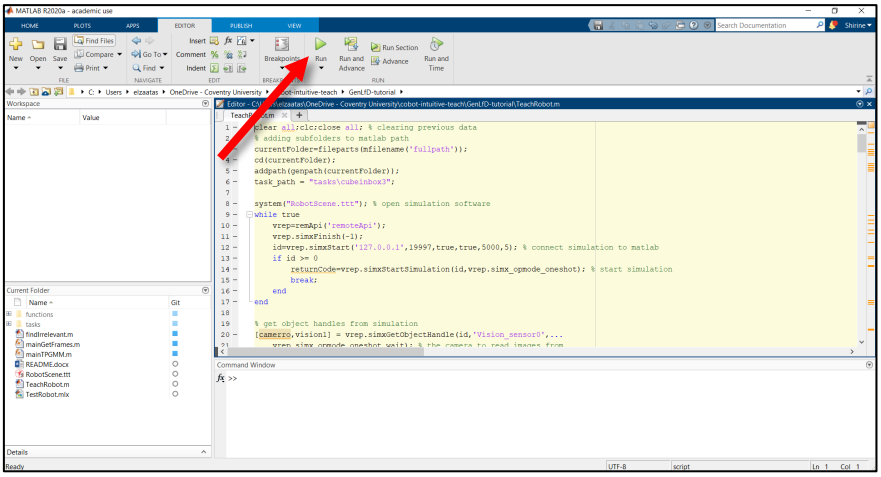
- TeachRobot.m, a MATLAB script used to record task demonstrations
- TestRobot.mlx, a MATLAB live script used to test the learnt task model

Firstly, open TeachRobot.m with MATLAB R2020a.



Step 2

When MATLAB launches, run the TeachRobot script by clicking Run, in the Editor Toolbar.

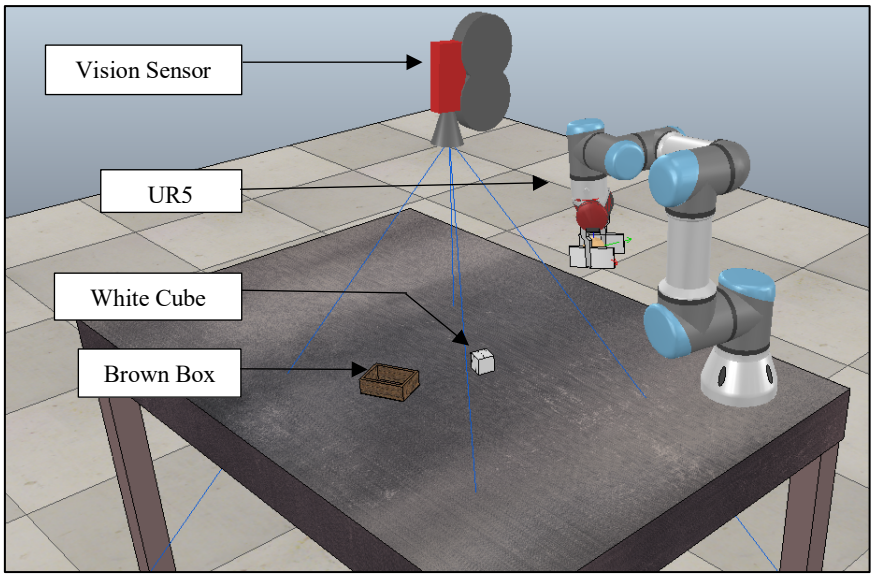


Step 3

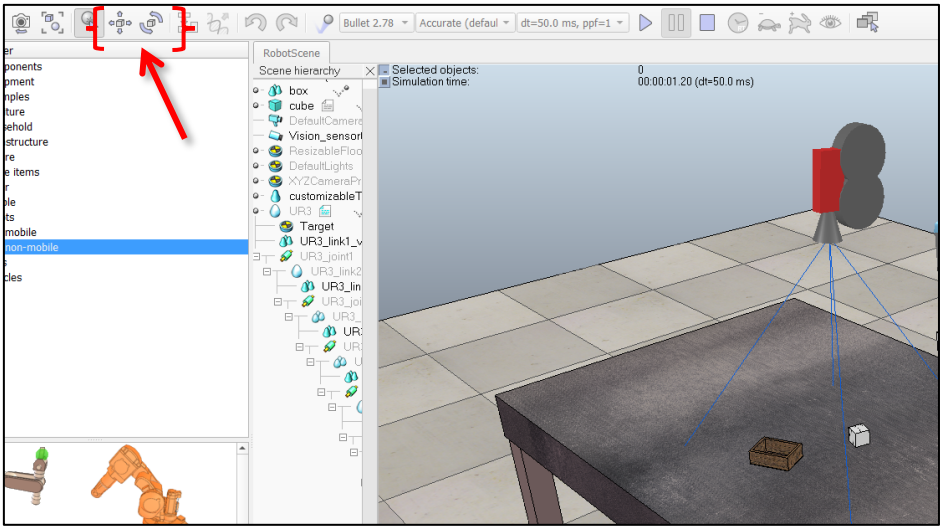
CoppeliaSim EDU simulation software will automatically open to a scene showing, a UR5 robot, a white cube, a brown box and a vision sensor.

The simulation will be used to record 5 demonstrations of the robot picking the cube and placing it in the box. The vision sensor will record a 2D image of the initial table setup of each demonstration.

In each demonstration, the position and orientation of the cube and box should vary, so the robot learns a task model.

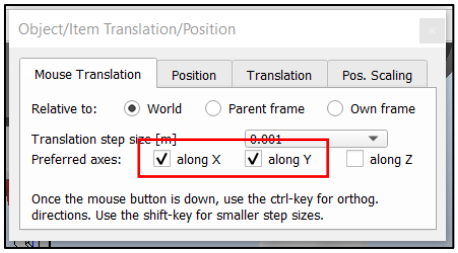


Use the object shift and rotate tools to vary the positions of **only** the cube and box on the table.

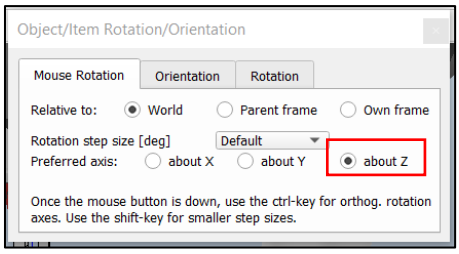


Make sure:

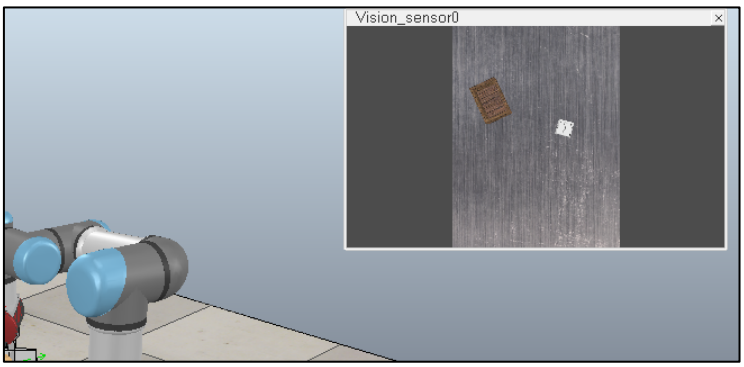
- You don't move the robot, the vision sensor or the table.
- When using the shift tool, enable shifting along the *x* and *y* dimensions only, relative to World.



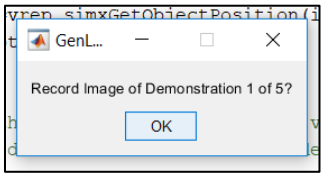
- When using the rotate tool, enable rotating along the *z* direction only, relative to World.



- The objects remain within the field of vision of the vision sensor. You can confirm this in the next step, or in the vision sensor's pop-up output.



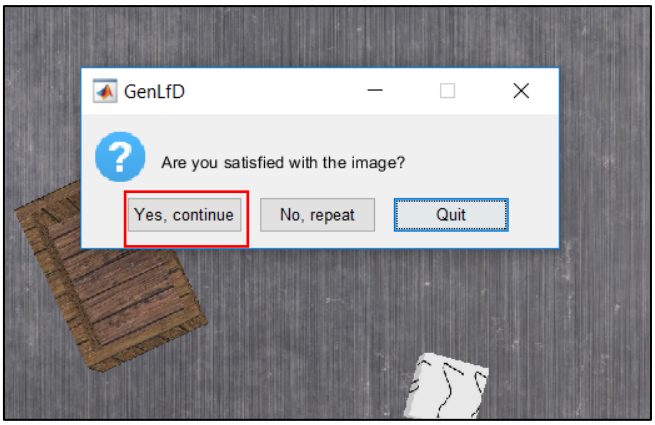
Once the cube and the box are moved and rotated to your choice, click *OK* on the GenLfD dialogue box that opens within MATLAB.



Step 4

The captured image will be shown as well as a GenLfD dialogue box asking you if you are satisfied with the image. Click *Yes*, continue if you are satisfied with the image.

However, if any of the objects are outside the camera's field of vision, move them again to your choice and then click *No, repeat*. If you would like to terminate the software, click *Quit*.

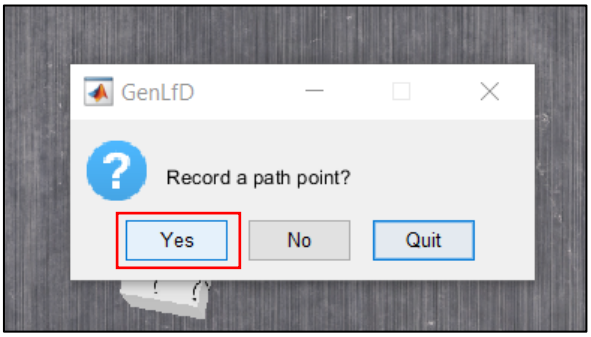


Step 5

If you clicked *Yes, continue*, now is time to record the demonstration path. In this task, each path is made from 4 path points:

1. Pre-grasp point, above the cube with open gripper
2. Grasp point, on the cube with closed gripper
3. Midway point, between the cube and box, above the table with closed gripper
4. Drop point, above the box with open gripper

In the next dialogue box, you will be asked if you want to record a path point. Click *Yes*, to begin recording.

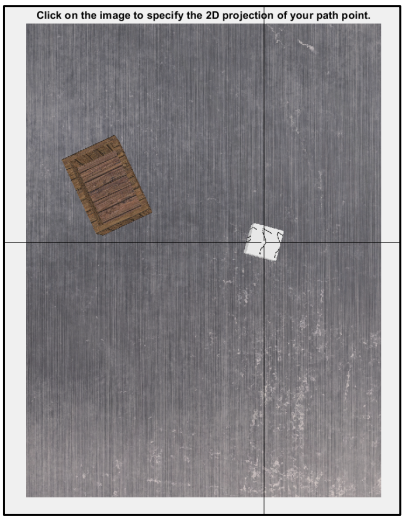


Step 6

The first path point in a pick-and-place task is typically above the object to be picked. In this case it is above the cube.

Each path point consists of 3 dimensions: *x, y* and *z*. Firstly, you record the *x* and *y* position by clicking on the cube in the image.

When clicking, observe how the robot end effector moves above the cube in CoppeliaSim.



A dialogue box will pop-up but **DO NOT** click it yet.

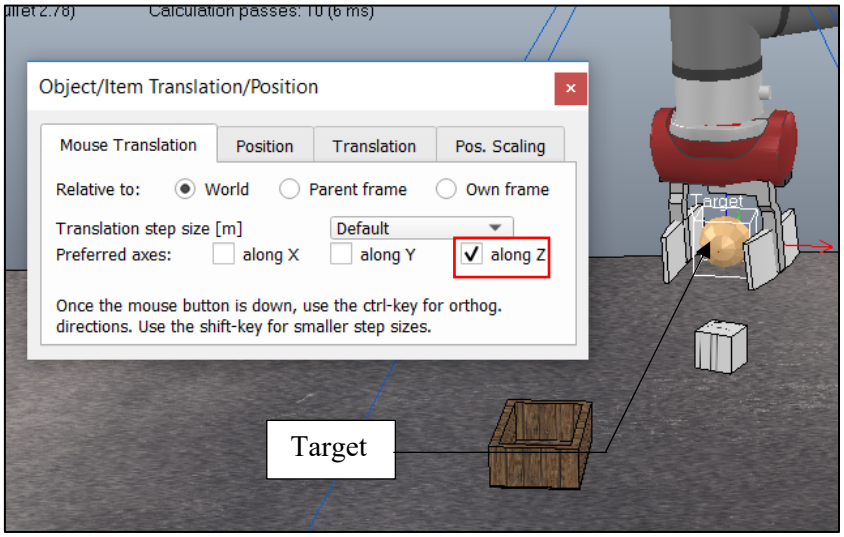
Step 7

In CoppeliaSim, you will find that the robot's gripper is now is above the *x-y* point that you chose in Step 6. You will see that there is a golden sphere within the robots grippers. This golden sphere will be referred to as the **Target.**

Now, you should adjust the z position of the path point, i.e. the height of the gripper above the object.

The **Target** is too high above the cube in the z direction.

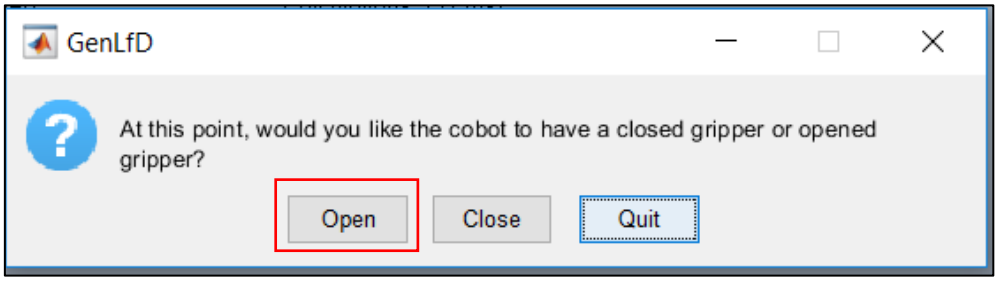
To bring the **Target** down to a more appropriate height, as in Step 3, open the shift tool then click on the **Target** and move it along the **z direction** down to a few centimetres above the white cube.



Return to the open dialogue box in MATLAB and now click *OK*.

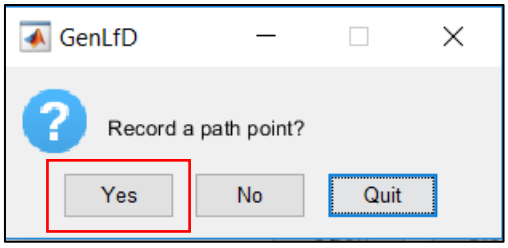
Step 8

After recording the coordinates of a path point, you are asked whether the gripper is to be open or closed at that point. For the 1st path point, click *Open*.

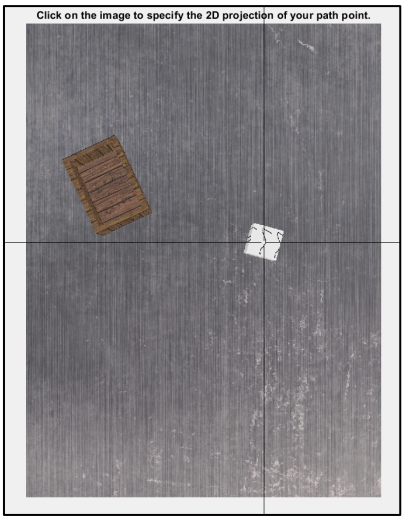


Step 9

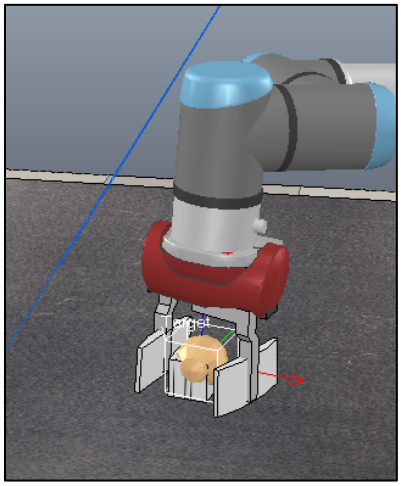
Next, you need to record the 2nd path point. Click *Yes* when asked to record a path point.



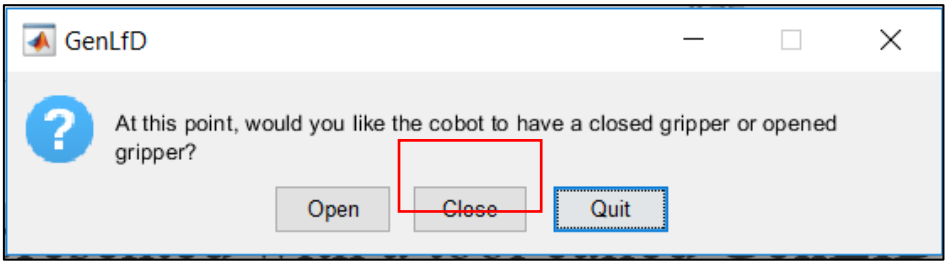
Secondly, click on the cube to record the *x*-*y* coordinates of the 2nd path point.



Then, adjust the height of the **Target** so that the cube is in between the gripper fingers.



When asked if the gripper is closed or opened, click *Close*. You will see the gripper will close in the simulation.

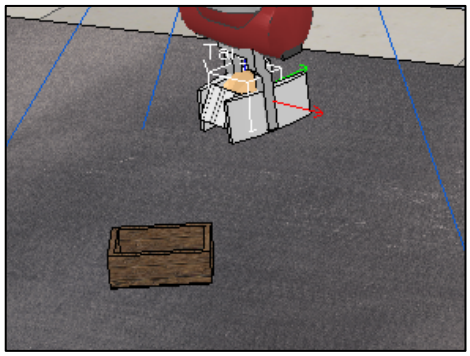


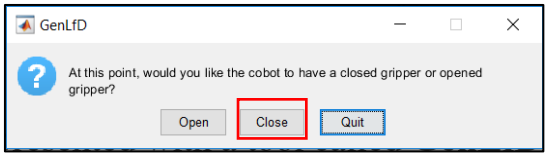
Step 10

Repeat the above step for the 3rd point; however, this time

- The *x*-*y* coordinates are between the cube and the box
- Raise the height a few centimetres above the table

- Keep gripper closed



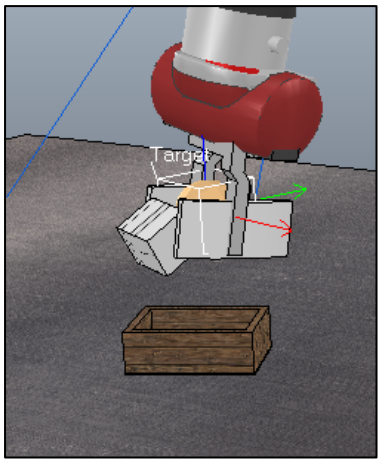


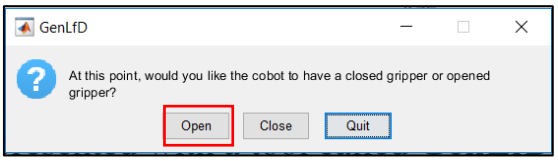
(Note: when the robot moves to the specified point, the cube might fall out of the gripper. This is just a simulation fault and will not affect the end learnt model. Therefore, continue the steps and ignore this fault. Refer to Potential Problems #1)

Step 11

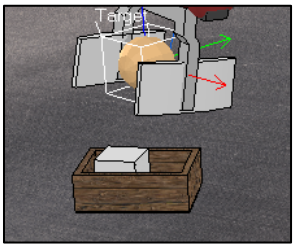
Repeat the above step for the 4th point which should represent the drop position.

- The *x*-*y* coordinates are on the box
- Adjust the height to be slightly above the box
- Set the gripper to open



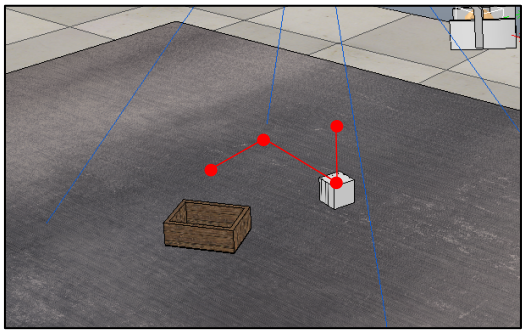


You will see that the object falls into the box and the task is completed.

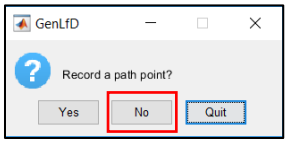


Step 12

You have thus recorded the necessary path points to accomplish the task.



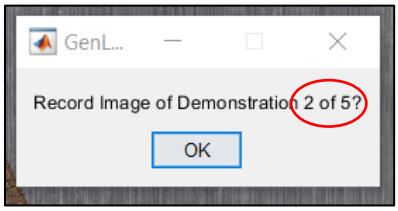
When prompted again to record a path point, click *No*, since the task is completed.



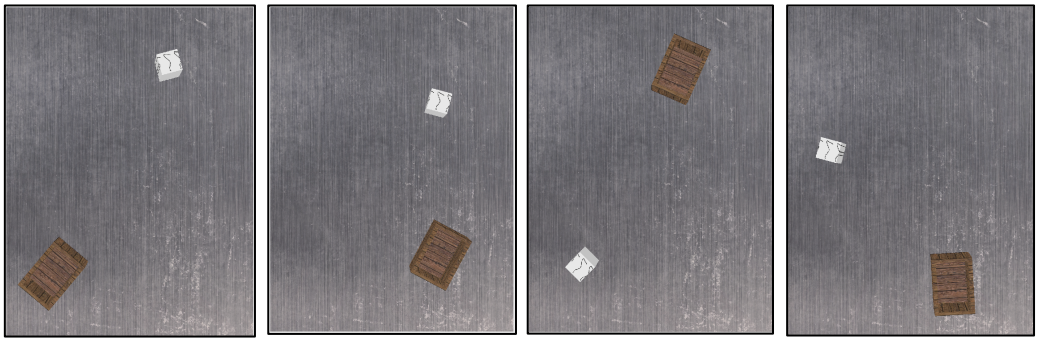
Step 13

For the robot to learn the task, you need to provide multiple demonstrations with varied positions of cube and box. That way, the robot will understand that the path points are dependent on the positions of the cube and the box.

Now you will repeat steps 3 to 12, 4 more times until you have recorded 5 demonstrations in total.



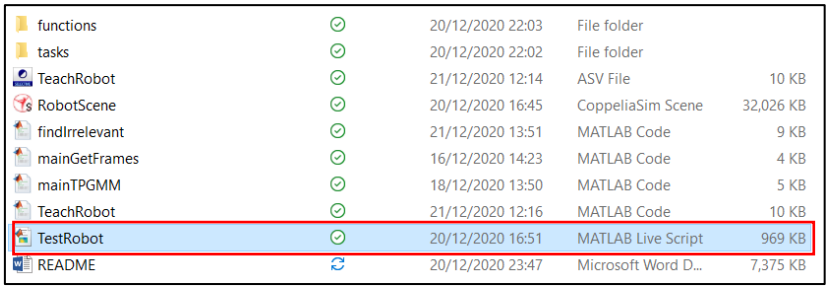
Make sure to vary the positions and orientations of the cube and the box well between demonstrations to create a good variety. Below are a few examples of how you can vary the positions of the cube and box.



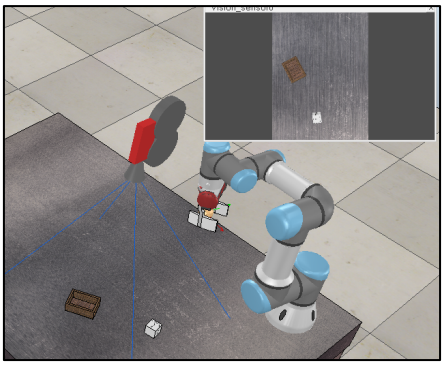
Step 14

Once the training is completed, you can test your learnt model.

In the folder, click on TestRobot.mlx.



In CoppeliaSim, vary the positions of cube and box such that they are in new positions the cobot has never seen.

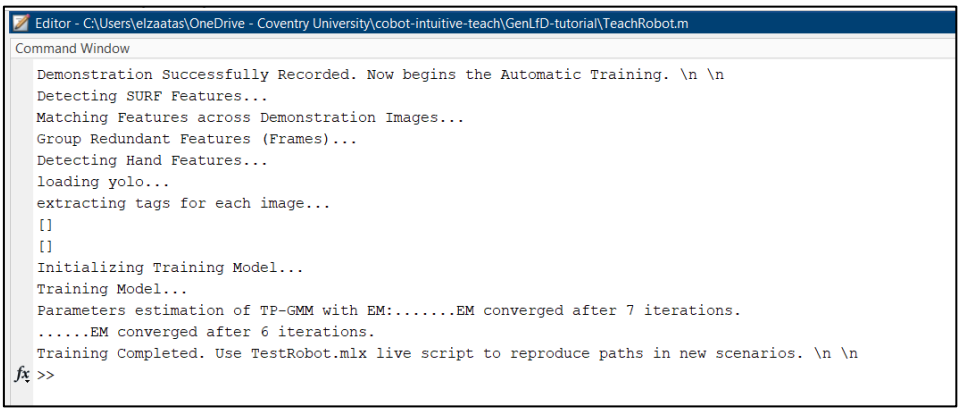


Run the TestRobot MATLAB live script. You will obtain the reproduced path in this new scenario.

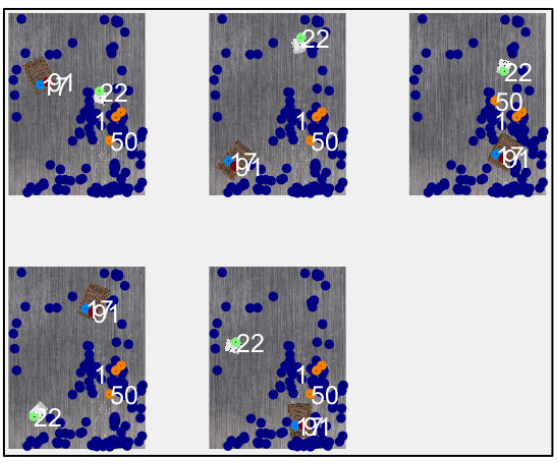
(Note: If you receive an error saying no subtask was matched, that means the algorithm failed to detect the

Step 15

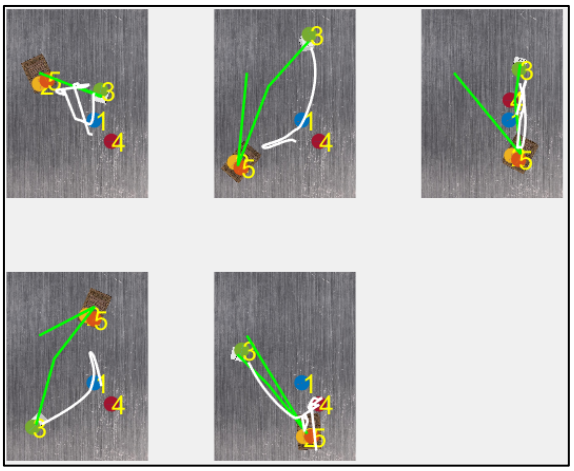
Once all the demonstrations are recorded, the training process starts automatically so you only need to wait. At the end, you will see the result of the training process.



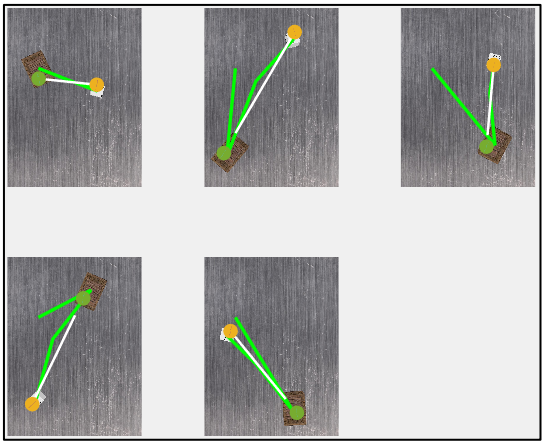
Firstly, the task parameters will be detected. They are the different dots in the image below. It is important that at least one dot belongs to both the cube and the box.



Secondly, the model will be trained to obtain probabilistic distributions of the paths with respect to the task parameters. Here we notice that the generated white path, is not satisfactory since it is very different than the ground truth path in green. The next step will solve this problem.



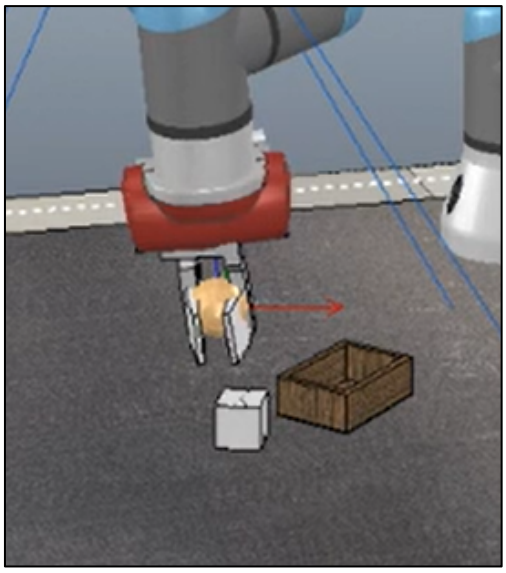
Thirdly, a reinforcement learning algorithm will identify optimal task parameters that generate a better path. You can see that the new path in white is much better than the originally generated path in red.



(Note: In the training process, two errors might occur. Refer to Potential Problems #2 and #3.)

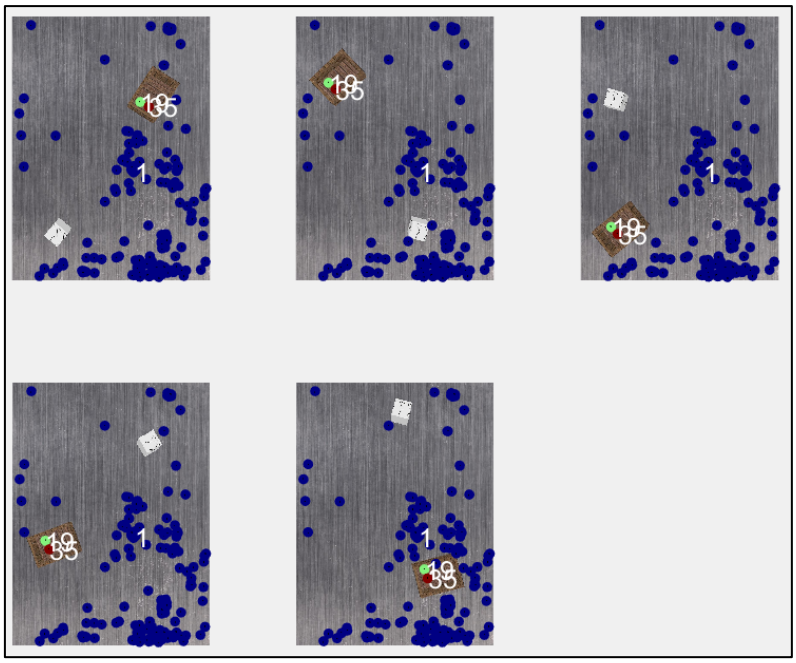
Potential problems

1. Failed Demonstration



When recording the demonstration, the cube might fall out of the gripper while moving. This is a simulation dynamics error and won't actually affect the algorithm performance.

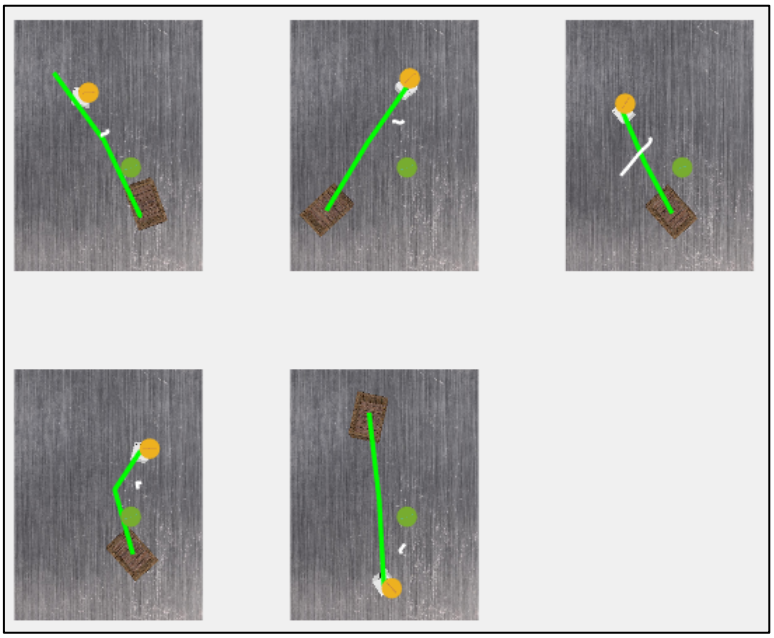
2. Missed objects



When looking at the matched frames of references (the dots), we can notice that none of them belong to the cube. That means that the cube was not detected by the algorithm. Therefore, the reproduced paths will be ineffective.

This could be because the object doesn't have prominent features, or it has a reflective surface, or there are shadows. Try to overcome the above problems and rerun the recording algorithm. If problem persists, seek help from the programmer.

3. Incorrect convergence



The reproduced path in white is unsatisfactory and not meaningful, that is because the green frame belongs to the table, not the box. When this happens, run the function of findirrelevant.m until satisfactory results are obtained.

4. No Features Detected

In TestRobot live script, the algorithm attempts to detect the task parameters in the new image. In rare occasions, they might fail to detect and in such a case, the path cannot be reproduced.

End of this Document