MASTER

Learning an Industrial Dross Skimming Task Using LfD Framework

Krishnamurthy, Sudharsan

Award date:
2020

Eindhoven University of Technology
Department of Mathematics & Computer Science

Learning an Industrial Dross Skimming Task Using LfD Framework

Master's Thesis

Sudharsan Krishnamurthy
1318039
s.krishnamurthy1@student.tue.nl

Version 1.0

Eindhoven, September 2020

Abstract

In recent years there has been a rapid progress in Industrial robotic systems. However, this also widened the scope and expectations of programming a robot. A robot should be easy to program and reliable in task execution. *Learning from Demonstration (LfD)* also known as *Robot Programming by Demonstration (rPbD)* offers a very promising alternative to mainstream approaches. We try to encode *dross skimming* task on an industrial robot using *Task Parameterised- Gaussian Mixture Models (TP-GMM)* - a type of probability based estimation LfD framework. A series of demonstrations are provided to the robot using tele-operation and are encoded using Gaussian distributions providing a method of spatio-temporal correlations. The trajectories are then generalized using Gaussian mixture regression. Finally we compare and analyse the computed dross skimming trajectory to the manual dross skimming and use this skill to generalize different contexts.

Index Terms - Task Parameterised- Gaussian Mixture Models (TP-GMM), Learning from Demonstration (LfD), Robot Programming by Demonstration (PbD), tele-operation, dross skimming.

Table of Contents

1. Introduction

1.1 Background and Project Context

As Steel is in high demand for various sectors such as automotive industry, electric household and civil, it is indeed necessary for large Steel and Iron companies, to produce high quality and low cost Steel. Therefore, a metric for the Steel to be of highest quality is to be anti-corrosive. This could be achieved by a process known as "*Galvanising*", where a continuous Steel strip runs through a molten Zinc coating bath. The chemical characteristics of Zinc (Zn) on Iron (Fe), makes the Steel able to withstand corrosion. The thickness of the coating is controlled by an air stream (also called as air knife). [1] The level of corrosion resistant is directly proportional to the thickness of the galvanising layer.

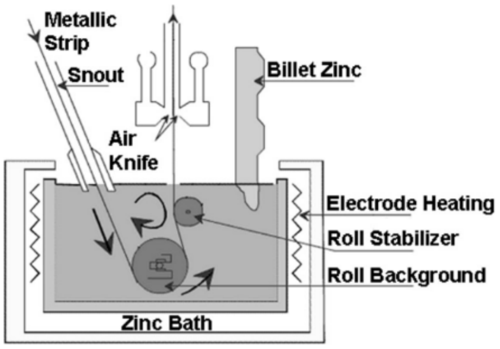


Figure 1.1: Continuous galvanizing Line. `Image Source: [2]`

When a Steel plate (or roll), gets immersed in the galvanising bath, it actively reacts with the molten Zinc and affect the physio-chemical characteristics of the bath. Fe-Zn (along with Al) intermetallics are formed at the interface between the coating and substrate. These intermetallics are hard and brittle and commonly known as *dross* [2]. According to [2] the effects of dross on commercial Zinc bath lines include, increase in the overall thickness due to Iron-Zinc compounds therefore degrading the outward appearance and quality. There is also a considerable wastage of pure Zinc in the form of dross which is undesired [3].

The dross formation is inevitable, that is one cannot prevent it. But there has to be a method to remove the dross from the active Zinc bath inorder to maintain the desired quality of the end product. Dross formation can be categorised as: Top Dross and Bottom Dross. Several commercial galvanising units remove the top dross by skimming. the bottom dross on the other hand is impossible to remove through skimming. However, one common solution is to convert the bottom dross into top dross by adding Aluminium (Al) into the bath. It is then skimmed without disturbing the galvanising bath. The concentration of Al has to be maintained at 0.12 wt.% (weight percentage) in order to avoid the reaction between Fe and Al and actively aid the desired reaction between Fe and Zn [4].

The dross removal is either done manually or using robotic solutions. In manual process, worker(s) suite up in protective gear, skims the dross from the surface on the Zinc bath. The worker(s) stand close by the hot Zinc bath which is usually 450 - 460 °C. Approximate weight of each dross scoop is around 10 - 15 Kg, which makes the task laborious. The worker(s) are also subjected to extreme working environments which causes serious health issues. Working very close to the hot Zinc bath with a risk of splashing, high risk of fire, Zinc dust causing breathing issues, bare minimum working distance from the hot Zinc bath posing high risk of injury, and continuous exposure to high temperature working environment [5]. Considering the safety of the worker(s), most of the industries deploy robotic solutions. Robots are considered as a potential alternate for performing tasks which are too laborious for humans, with high precision, accuracy and speed. Most of the industrial robotic solutions include a robotic arm with certain Degree of Freedom (DoF) which move accordingly perform several tasks. Robots could be completely autonomous or semi-autonomous. However, there could be a high risk of damages caused by an autonomous robot deployed in a (un)structured environment if not properly trained. When it comes specific to the application of robotic solutions for Dross skimming(removal), one of the method is the use of tele-operated semi autonomous robotic arm as it can deal with several uncertainties in the environment. [6].

Heemskerk Innovative Technology (HiT) is currently working on an approach to apply Artificial Intelligence (AI) to a dross skimming robot. The robot can be deployed at an industry which will skim the dross from the Zinc bath. If done, the robot could replace human workers in such challenging environments. Currently, a simulated robotic arm has been modelled which is controlled by a haptic device. The device drives the robotic arm (in simulation) which aids the dross skimming process. HiT wants to move a step ahead and develop policy driven solution for the robot, which (semi-)automates the dross skimming process.

1.2 Robotics for Industrial tasks

1.2.1 Robot Programming

Industrial robots are generally designed, (pre-)programmed to attain a specific task. Depending on the variations in the task environment, corresponding alterations are made to the robots by re-programming them. However, the complexity of programming remains one of the major hurdles preventing automation using various industrial robots [7]. This is usually done by software programmers who must have prior knowledge of the robot and it's environment and various other interactions. However, with the increase in task complexity, the time required to program the robot also increases. In an ever changing world, this becomes far more impossible on the longer run. Also once deployed, further alterations on the robot can be made only by those experts who have domain knowledge about the robot. While considering all these, manual programming doesn't seem like an ideal solution[8].

1.2.2 Robot Learning

An alternative is to enable a robot to autonomously discover and learn an optimal behaviour through exploration by trial-and-error interactions. This method is known as Reinforcement Learning (RL) as this deals with providing feedback to measure the robot learning performance instead of explicitly detailing the exact solution to the problem [9]. The robot explores the environment and learns by itself without an expertise to program the tasks. In an industrial environment it is quite difficult for the robot to attain a convergence point to an optimum learned policy, considering the complexity of the environment and various other parameters [10]. It takes a lot of time for the robot to get trained in order to attain a generalised policy. During this time, the probability of certain undesired decision taken by the robot is high, which could cause potential damage to the robot or the environment itself. The risk could be avoided by training a robot in a simulated environment. In a simulated environment, we do not have a proper knowledge on all task and environment related factors. Because of this unavoidable differences in the simulated and actual task environment, the robot could show undesired behaviour. The behaviour could be corrected in the later stages but the risk of damage involved is always a concern. Hence, considering the time and risk factors involved, RL approach is also not a desired solution to train the robot for industrial specific tasks.

1.2.3 Robot Teaching

From the previous sections, it could be inferred that manual programming and autonomous learning are not the optimum solutions/ways to achieve a task completion using robots. Fortunately, an alternative technique could be Robot Learning from Demonstration (LfD), also known as Programming by Demonstration (PbD) [11] [12]. The main idea behind LfD is that to train or make the robot learn from demonstrations. These demonstrations are in the form of dataset which are essential for deriving policies, where the examples (dataset) are gathered using several demonstrations. LfD within itself has various methods for demonstration and is considered as a sub-sect of Artificial Intelligence [13]. The robot maps the recording of the demonstration and tries to mimic. There are 2 categories through which the robot can accumulate knowledge. One method is through *"Direct"* mapping in which the trainer operates the robot platform and the robot learns through it's own movements. The second option is known as *"Shadowing"* technique in which it records the data through various sensors on the trainer and tries to map and mimic [11]. The main advantage of LfD is that, it doesn't require much of domain knowledge or technical insights about programming the robot which makes it even more simpler and efficient for common people to train the robot.

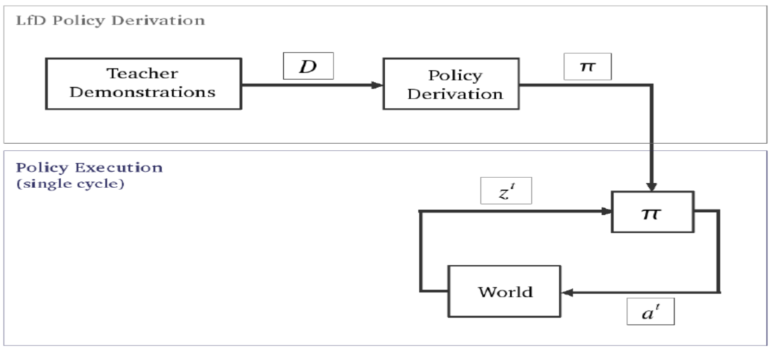


Figure 1.2: General Block Diagram of LfD
`Image Source - [11]`

From Figure 1.2 LfD can be categorised into 3 main process. A demonstration, a policy derivation using low-level encoding techniques, and finally a task execution or replication by the robot in the real world. An in-depth study on various evaluation factors and low-level encoding techniques are presented under Chapters 2 and 3 respectively.

Demonstrations can be done in several ways. Such as, *"Teach box"*. This is a portable, programmable controller which enables the controller to examine the state and drives the robot [14]. This approach is quite tedious since it requires some pre-programming of the teach box according to task specifications [15].

Another common method of task demonstration is using Kinaesthetic teaching, where the human teacher or trainer demonstrates the task using the actual robot as a tool. The robot records the actions and parameters according to the movements initiated by the trainer. This method produces the best results in terms of simplicity, time factor and efficiency [16]. However, there are some disadvantages such as, the inability of the human trainer to Kinaesthetically teach the robot in hazardous environments (which is exactly the case to be considered for dross skimmer, since the environment is hazardous to work and also the size of robot is large making it practically impossible to carry out Kinaesthetically teaching). Additionally the demonstrator- induced dynamics during the training stage, might alter the dynamical conditions between the demonstration and the autonomous stages [17].

A third demonstration technique in which a robot is operated by the trainer via a joystick (haptic device) from a (not necessarily) remote location. Both the Kinaesthetic teaching and Teleoperation are categorised under *"Direct"* method of knowledge transfer from trainer to robot [11].

The teaching can be done remotely which is actually an advantage in Dross Skimming operation. One challenge in LfD using Teleoperation should be manageable [11]. That is, low-level motion demonstrations are difficult on systems with complex motor control (such as high degree of freedom robots). Further more, Teleoperation is limited due to the spatial-temporal variations, compared to other demonstrations methods [18]. A good learned policy depends on the quality of teleoperated teaching, or in other words, the success of teaching process is highly dependent on the experience of the demonstrator.

1.2.4 Interaction Environment

A fundamental factor to be considered while dealing with the interactions of robot with the environment is, the task specific parameters. Depending on these factors, certain manipulations of the robot should be taken into consideration, such as, the contact force, velocity etc. In case of dross skimming, the task environment is considered as structured and predictable, in which the external disturbances are less, it could be said that the task parameters are much related to the task itself rather than the surrounding environment. For example, the robot for dross skimmer is positioned at a particular spot and bath dimensions, extrudes, steel strip, ingot could be considered as task parameters. Whereas when certain robots are deployed in an unstructured environment such as a care robot in a hospital, it is expected from the robot to interact in dynamic objects. Therefore, while training a robot in an unstructured environment, dynamic objects should also be considered as task parameters. optimal logical solution within this dynamic environment is quite hard. In such cases a more strategic level of learning is required rather than an operational level. Given the fact that, CGL is structured environment, an operational level of model learning could be logical. This could yield a fundamental solution that does the task of autonomous dross skimming. An additional step to this could be more strategic approach to incorporate the dynamic behavior of dross formation while learning.

1.3 Motivation

Several challenges exists in manual dross skimming operation such as, the safety concerns of human workers working in the proximity of the Zinc bath, the time and complexity involved in manual programming of robots, and learning a task using RL. Based on these challenges, the motivation for this research assignment can be formalised as follows:

Considering the importance of worker safety, time involved, ease of programming and reliability, robotic autonomous dross skimming operation using LfD has a good scope of research as it can serve as a potential alternate to problems involved in manual dross skimming and other mainstream robotic solutions.

1.4 Problem Statement and Goal

Due to the difficulties in the working environment in commercial galvanising lines, extra care should be taken by the worker(s) in order to prevent serious hazards. Inspite of using certain safety measures, serious health issues are inevitable in the long run. So, to prevent the exposure of human workers to such hazardous environment, a robotic dross removal solution is investigated. If such a solution would prove successful in terms of performance and efficiency, it could also help in speeding up the process of Galvanising in CGL. To achieve this, the primary milestone is to mimic the dross skimming task done by a human worker by hand. Since the CGL runs for 24/7, developing and testing a robotic solution on actual line is not (yet) a feasible approach. Therefore, a simulated setup of CGL with tele-operation is considered.

The goal of this assignment is to develop a simulated autonomous dross skimmer by extracting the motion profiles demonstrated by the operator and encode these motion profiles into the robot. The generated or learned profile is then compared with manual dross skimming operation. That is, the autonomous execution is assessed with the manual dross skimming using the performance parameters considered in this assignment. The assignment is more focused on the operational level of learning so that the importance is given to extracting the movement profiles of dross skimming which is similar to human worker performing the same task by hand and less concentrated on the strategic level. That is, the dynamic behavior of the dross is not considered in this assignment such as tracking of dross and choosing a particular strategy according to different dross formation while executing the trajectory.

1.5 Research Question

Based on the Problem Statement and discussed in Section 1.4, the research question is formulated as follows:

1. ***How the skills of an expert can be incorporated in automating dross skimming using Learning from Demonstration with a tele-operated haptic device in a simulated environment?***

   (a) ***What are the important factors associated with the Zinc bath and the robot aiding the skimming process?***

   (b) ***Which low-level encoding techniques can be utilized for learning dross skimming operation?***

   (c) ***How to build a policy driven platform for dross skimming with Unity in a simulated environment?***

   (d) ***How the key performance factors could be validated with respect to autonomous dross skimming in a simulated environment?***

1.6 Approach

To achieve the overall goal previously stated the approach will be as follows: Initially a new dross skimming setup is developed with an ABB industrial robot in *Unity* simulator. The task is demonstrated using a tele-operated haptic device called "*Geomagic Touch*". The task is then encoded using a python based TP-GMM model. The generalised trajectory obtained from the model is then executed in the same Unity setup.

1.7 Report Outline

This section presents the outline of this thesis. Chapter 2 describes the parameters with which the dross skimming is evaluated. Additionally, it presents an overview of the parameters related to Zinc bath and robot separately. Due to the lack of knowledge on some factors, they are either assumed to be constant or ignored during the course of implementation. Chapter 3 discusses various low-level encoding techniques under LfD framework and are analysed with a set of requirements. A suitable algorithm is chosen to encode the dross skimming task based on the evaluation of each algorithm with these set of requirements. Chapter 4 proceeds with the implementation of dross skimmer using TP-GMM model in a Unity simulator. The demonstration, dataset preparation, learning and trajectory extraction are detailed under this chapter. Finally, in Chapter 5 conclusion are drawn based on the experiments conducted with respect to the performance of autonomous dross skimming. Moreover, recommendations are suggested for future work.

2. Performance Indicators for Dross Skimming

In order to create a policy for any application using Artificial Intelligence, before defining the algorithm, one has to have a complete idea on what factors should the model work. In other terms, depending on a number of factors, a model has to be implemented and evaluated. These factors are more specific based on the application. For example, a robotic pick and place task has specific factors to be considered whereas a robotic dross skimmer has certain other requirements. Starting from the task environment to the type of robot used everything differs. These could be considered as evaluation factors/ parameters for that particular application. In this section, we would be discussing about these factors that are related to dross skimmer. Based on these factors, the final implementation would be evaluated. This chapter is divided into two sections: in which the parameters related to Zinc bath and those related to the robotic arm discussed separately. Although, these two go hand-in-hand, categorising them would lead to a broader prospect of including as much factors as possible. Since there is no international standard set in designing a dross skimmer, it is upto the industry to determine the bath and robotic arm specifications. Therefore most these parameters are based on the visual inspection and analysis rather than supported by literature. The sub-question that will be answered here is:

1. (a) ***What are the important evaluation criteria associated with the Zinc bath and the robot aiding the skimming process?***

2.1 Parameters Related to Zinc Bath

- Dimension of the bath

  Initially, the most important thing to start with is the dimension of the bath. A robot must know it's limits of dross skimming. These limits correspond to mapping the dimensions of the Zinc bath in x, y, z plane. A 3-D mapping is sufficient as this should remain constant for a particular bath. During the course of this assignment, the dimensions of the bath are set constant. As there are no specific standards in defining the dimensions of the bath, different industry can have varied dimensions of the bath. It is therefore necessary for the robot to know the exact dimensions of the bath so as to train the model for the task skimming task. For this experiment, the dimensions are as per the bath design specified in Figures 2.1 and 2.2.

- Level of Zinc

  For a robot to have a reference on removing the top dross, it has to know the level of Zinc in the bath. Since the steel sheet moves continuously inside the CGL, considerable amount of Zinc is removed from the bath over time. Therefore, the level of Zinc is varying in the y-axis. A constant monitoring is required inorder to provide a reference to the robot. Failing to do, the end effector will either skim air or submerge fully insides the Zinc bath (both of which are completely undesired). Figure 1.1 represents the y-plane representation of CGL which represents a much clear depiction of the importance of determining the level of molten Zinc using level sensors. However, for this implementation we consider the level of Zinc to be constant and train the robot.

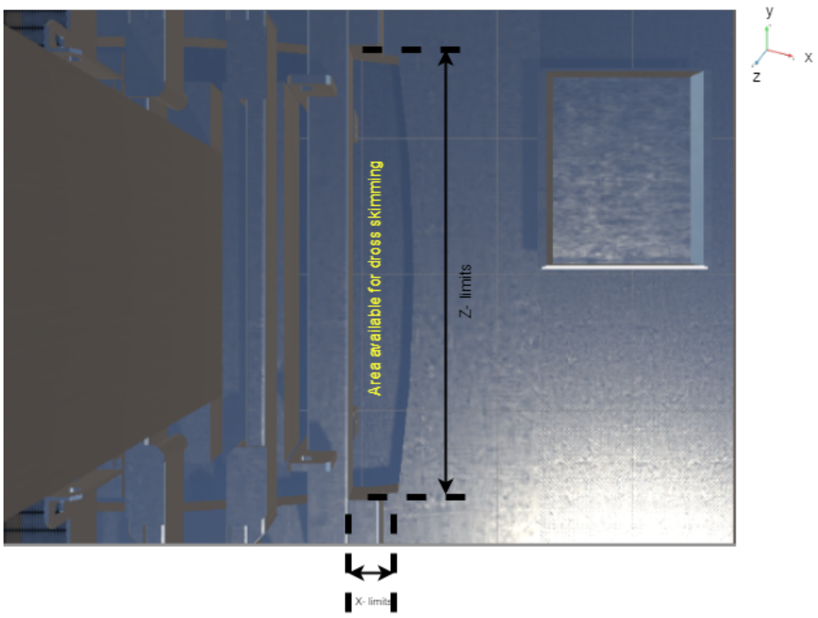


Figure 2.1: This Figure represents the top view of the CGL on x-z plane. The area available for dross skimming is bounded within x and z limits are highlighted here. For this experiment, these dimensions remain constant and the model is trained with reference to these dimensions.

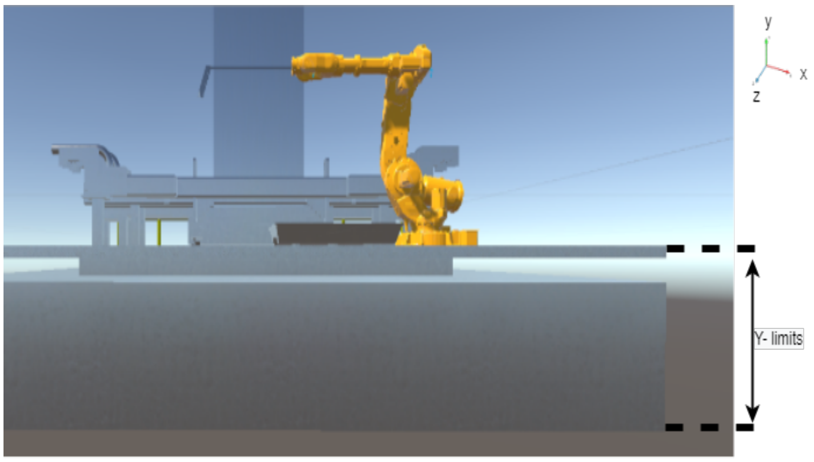


Figure 2.2: This Figure represents the side view of the CGL on y-z plane. Here the level of molten Zinc in the bath is bounded within the y-limits.

- Disturbances in bath during skimming

  If, during skimming, the robot creates a lot of disturbances (in the form of waves) in bath, the incoming steel plate gets inconsistent galvanised coating and creates pimples on steel strip (dross getting coated on the steel strip). Also dross might get coated along with pure Zinc. If such a scenario arises, the robot should sense this and decrease the contact force and velocity of dross skimming has to be adjusted at an optimum point. Since there is no possibility to temporarily stop the galvanising process, the optimum velocity has to be adjusted without interrupting the galvanising process.

- Periodicity of dross removal

An analysis over a period of time has to be done to determine the mass of dross. If the quality of end product has more % of impurities in form of dross, the frequency of dross skimming has to be increased to remove as much dross as possible. With this estimation, one could also analyse the number of skimming operations required for a period. It should be noted that, the dross skimming process is not continuous.

2.2 Parameters Related to Robotic arm

While considering the factors related to robotic arm, these are directly associated to the specifications of the industrial robot being deployed in the task environment. Replacing a robot must make sure to re-consider all the factors.

- Velocity of dross skimming

  As discussed in previous section 2.1, a disturbance in the bath is created if the robot approaches the bath with a greater velocity. Doing so, the disturbances might cause inconsistent coating of Zinc and reduces the quality of the end product. The lower the velocity, the efficiency of dross skimming is too less as there is a tendency of dross getting solidified. This often results in dross sticking onto the end effector increasing the payload of the robot. A balance has to be achieved between these two conditions as: the dross skimming shouldn't introduce disturbances in the bath and the efficiency of dross skimming is maintained at optimum levels. It is relatively much easier to vary the velocity of the robot to find this balance.

- Amount of dross skimming for a period

  This goes hand in hand with the bath parameters. An estimation on the dross volume could be extracted in simulation by using a Unity package: "*Flex*" for this project. In practical application dross skimming, if traces of dross in the end-product, additional skimming has to be performed to minimize the % of dross. This feedback is not obtained immediately. After galvanizing, the steel roll goes through several stages. One such (much later) stage is quality inspection. The quality control team has to give feedback on the % of dross found. So that, necessary changes could be made in the skimming operation.

- Collision avoidance

  It is expected from an autonomous manipulator to detect and avoid collision in the task environment. A collision with CGL could result in serious damage to the bath/ steel strip or to the robot itself. To avoid such undesired situations, a collision avoidance safety mechanism has to be incorporated within the robot system so that no collision occur. Even if one has to occur, then the robot has to stop it's autonomous execution.

- Trajectory deviation

  The human worker has to be continuously notified with the robot's trajectory in real time in order to see whether the desired operation is performed by then robot. If not, then necessary corrections has to be made to the learning model through additional demonstrations or by adding task parameters. The deviation boundary surrounding the bath serves as a reference for the robot to check for it's deviation. If the end-effector violates any such boundary condition, the human worker has to be notified about the trajectory deviation.

3. Low-Level Encoding Techniques

This chapter provides an overview of the available low-level encoding techniques. An encoding technique refers to capturing or encoding the demonstrated data (in our case, the skimming operation) by creating a state-action policy. A policy might contain several state-action pairs. Based on this, a policy finds an appropriate action depending on the current state. An encoding technique can be categorised as follows: a*" low-level representation"* where a nonlinear form of mapping between the sensor and motor information are taken. A *"high-level representation"* follows encoding the skill into a sequence of action-perception units [19].

The robotic arm has to function based on all the evaluation parameters discussed in Chapter 2. Therefore, how various encoding techniques can learn a policy based on all these evaluation factors will be addressed by the sub-question:

1. (b) ***Which low-level encoding techniques can be utilized for learning dross skimming operation?***

A generalised policy in LfD is derived based on the demonstration data. However, some methods do not require demonstration data to derive a policy. These include, Reinforcement Learning (RL) where the robot the environment through exploration or trial-and-error. Although, RL can be used where in the robot explores the environment, chances of undesired results raises a safety concern among industries. Fearing of damages caused in the environment due to exploration, an alternative solution is to simulate the environment and train the robot. The results achieved are only as good as the environment used in simulation. It also depends on the differences in mapping between the simulation and real world (either in terms of the environment or the manipulator itself). For all these reasons, a pure Reinforcement Learning approach for Dross Skimming has not been adopted for Dross Skimming. Therefore initially, a Human-In-the-Loop (HIL) approach wherein a human demonstrator teaches what has to be done by the robot using a tele-operated device is analysed.

On the other hand, recent developments from [13] talks about Deep Reinforcement Learning, where an approach to train the robot based on human demonstrations. When a trained model has been achieved, it is only as good as a human demonstrator. Here is where, Reinforcement Learning has been utilized to improve the robot performance using some reward functions. The reasons for adapting this method could be categorised as a drawback for using Reinforcement learning alone. Which is, too much RL could result in overload of states producing undesired results and also the training time [20]. So first the skill is learned from human in simulation using tele-operation and RL is further used to optimize or refine the skill. The human demonstration provides information how to perform dross skimming. And additionally use RL to improve or optimize the dross skimming process. Because certain adaptive methods has to be utilized to further optimize or make corrections with the inaccuracies in robot model description in simulation.

This chapter analyses few methods for learning the skill (behaviour). A common distinction between trajectory-based which uses time as a function for motions, and state-based use states of robot for motion is discussed extensively in [21]. But this chapter focuses mainly on two distinctive categories:

- Probabilistic Methods
- Deterministic Methods

as the probabilistic methods outputs the variances from the observed behaviour while in a deterministic model, the actual goal is determined (always the same without no output variances) based on the input behaviour.

3.1 Requirements for Low-level Encoding Techniques

This section describes a set of requirements that provides an idea on comparing and capturing important characteristics of various low-level encoding techniques. The set of requirements are categorised according to the task execution and learning behaviour. Based on these requirements, a cost function could be derived determining the success rate of the learning policy on applications. The requirements are generalised as it can be adapted for any practical application.

3.1.1 Requirements for Task Performance

- Application
  - The learned policy should be suitable to handle various types of tasks encountered by the robot. For example, a dross has to be scooped using different scoop angles, using different velocity and force, adapt to the changes in the bath level.
- Demonstrations
  - The number of demonstrations required to learn a skill. A smaller set of demonstrations is favorable since this will reduce the time required for learning and more economically feasible in practical applications.
- Parameters
  - Parameters or Constraints should be able to be integrated into the learning framework. For instance, a complete movement profile should lie within the range of the robot. Choosing a task parameter for learning which is not within the robot range would be meaningless.
- Generalisation
  - The techniques should be able to generalize well such that they are able to handle new conditions for motion behavior. For instance, the references or task parameters for the dross skimming are the dimensions of the bath. If there might occur a scenario, where the dimension of the bath can change. In such a case, the technique should adapt to the new task parameters.

3.1.2 Requirements for Skill Learning

- Scaling
  - Scaling or more specifically Temporal Scaling refers to the ability of the method to be sped up, or slowed down during the autonomous execution phase.
- Robustness
  - No dataset is free without noise. A good encoding method should be able to cope up with noisy data from sensor recordings which often contain stochastic signals or missing data.
- Encoding and Regression
  - A technique discussed in [22] and [23] uses an optimisation technique called *"Dynamic time Warping (DTW)"* for encoding and regression. A requirement for a Skill Learning is to simultaneously perform encoding and regression without DTW (or other techniques).
- Incremental Learning

- A type of learning method is Incremental/ Online Learning [11] [24] where the operator is allowed to make alterations to the demonstration/ learning process. The encoding technique should allow the human in the loop to perform such modifications.

3.2 Algorithms

This section discusses some of the Deterministic and Probabilistic low-level encoding techniques in order. For each method, the algorithm will be reviewed highlighting some important advantages and limitations. The final section 3.3 of this chapter discusses how these techniques meet the requirements (as in Section 3.1).

3.2.1 Dynamic Movement Primitives (DMP)

The idea behind Dynamic Movement Primitives (DMP) is finding a method for adjusting complex motor actions without manually tuning the associated parameters. A dynamic system with a specific stable behaviour can be modulated with nonlinear terms. The goal is to achieve the desired point attracted behaviour using modulation with the nonlinear terms.

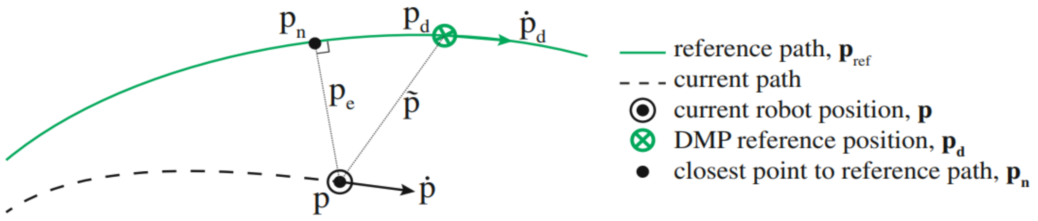

p_n
p_d
$\dot{p}_d$
p_e
$\tilde{p}$
p
$\dot{p}$
reference path, $\mathbf{p}_{ref}$
current path
current robot position, $\mathbf{p}$
DMP reference position, $\mathbf{p_d}$
closest point to reference path, $\mathbf{p_n}$

Figure 3.1: Illustration of a position tracking mechanism in DMP using position variables. `Image Source:` [25]

This can be represented using first-order systems as:

$$\tau \dot{z} = \alpha_z(\beta_z(g - y) - z) + f(x) \tag{3.1}$$

$$\tau \dot{y} = z \tag{3.2}$$

Equations 3.1, 3.2 look very similar to a damped spring model with an additional forcing term. The system is made to be critically damped with $\beta_z = \frac{\alpha_z}{4}$ so that y could monotonically converge towards the goal g.

Where,

- τ is a temporal scaling constant
- α_z and β_z are positive gains
- g is the goal point
- y and z are the state variables

DMPs have some interesting properties of generation of Kinematic movement primitives. One such and important advantage of DMPs is that it is possible to train with just one sample profile since the system is deterministic and time invariant. Thus allowing a better control to speed up or slow down the policy learning. This properly of DMPs allows us to use increment/ online learning since the temporal scaling allows the synchronisation of learned profile and training data. To realise this, in [26] online learning was implemented using an phase estimator which, depending on the output of the human tutor changes the speed of sawing(the example taken by the authors). Based on the human tutor, the phase and state estimator adapts the learning policy of motion and speed by identifying whether the current demonstration is very close to the previously demonstrated data. When the demonstration is close, the robot is treated as a leader increasing stiffness. When the demonstration deviates more, the tutor is the leader with normal path and force. After a few demonstrations with online adjustments, progressive automation with few demonstrations can be realised [25] [27].

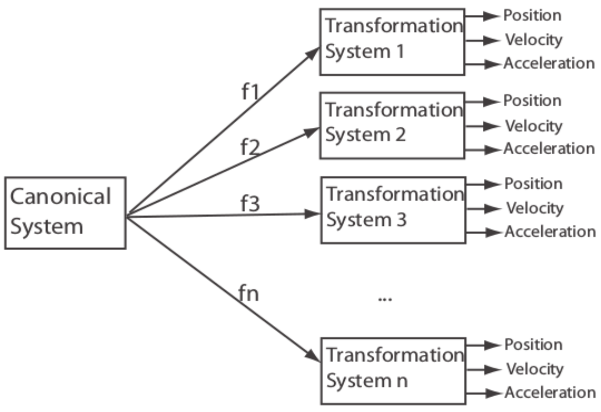


Figure 3.2: Illustration of A single canonical system with several independent function profiles. Normally this is the case with any robotic system. A typical robotic system has multiple DOFs. This figure depicts that each DOF has it's own forcing function and trajectories(transformations) `Image Source:` [28]

The ability to generate movement primitives using DMP is easy and quick. However, adapting DMPs to new task requirements becomes difficult when the manipulator parameters donot provide a trajectory corresponding to the parameters associated with the actual task [29]. Consider a Zinc bath with certain dimensions where the task parameters corresponding to the joint space parameters of the robot. When the dimension changes, the joint space parameters no longer makes sense and DMP encoding produces non-desired behaviour. Therefore, the trajectory profile is not much generalised in DMP when the environment is scaled implying the importance of task associated parameters. A common approach to tackle this is to generalise the demonstrations by performing it several times and an average of all these is applied to the new environment. This contradicts the advantage of DMP as discussed in previous paragraph where it was summarised that DMP requires only one demonstration.

3.2.2 Probabilistic Movement Primitives (ProMP)

A framework development in which the Movement Primitives (MP) is formulated using a probabilistic approach was introduced by Paraschos et al. in [30] as Probabilistic Movement Primitives (ProMP). Unlike DMPs, the advantage of using ProMP is that it is a probabilistic approach thus allowing to encode variance in some linear systems. On comparing ProMP with DMP, the former allows to determine a relation between different movement profiles, incorporates the temporal modulation needed for speeding up or slowing down the execution of Rhythmic and Discrete movements.

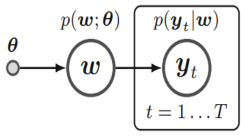

$p(\boldsymbol{w};\boldsymbol{\theta})$
$p(\boldsymbol{y}_t|\boldsymbol{w})$
$\boldsymbol{\theta}$
$\boldsymbol{w}$
$\boldsymbol{y}_t$
$t=1\ldots T$

Figure 3.3: This figure represents a Hierarchical Bayesian Model. The probability of observed trajectories $p(y_t \mid w)$ depends on the weight vector $p(w \mid \theta)$. `Image Source:`[21]

$$y(t) = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \Phi_t^T w + \epsilon_y \tag{3.3}$$

$$p(\tau \mid w) = \prod_t \mathcal{N}\left(y_t \mid \phi_t^T w, \sum_y\right) \tag{3.4}$$

where,

- Φ_t is a vector of dimension 2 x n = $[\phi_t, \dot{\phi}_t]^T$ for joint positions q_t and velocities $\dot{q}_t$
- n defines number of basis functions
- $\epsilon_y \sim \mathcal{N}(0, \sum_y)$ represents zero-mean Gaussian noise with $\sum_y$

The variance is computed with the weight vector w using a distribution $p(w;\theta)$ is introduced

- Parameter $\theta = \{\mu_w, \sum_w\}$

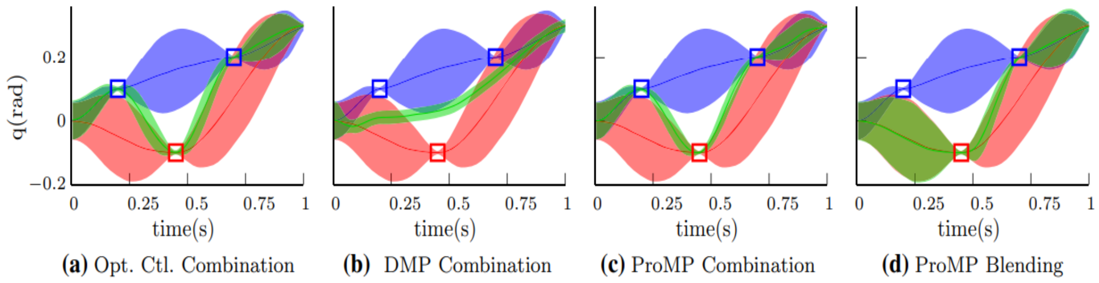


Figure 3.4: Comparison between two primitives. Combination of 2 movement primitives (shown in red and blue) and the resultant in green. **(a)** shows the optimal behavior of distribution obtained by adding both cost-functions. **(b)** shows combining DMPs which results in a linearly interpolated trajectory. The resultant misses all the via-points. **(c)** Co-activation of two ProMPs results movement passing through all via-points. **(d)** smooth blending from red movement primitive and, subsequent switching to follow blue movement primitive.
`Image Source: [30]`

While on the upside, ProMP encodes variance of the trajectories which was not possible in DMP. With the option of encoding variance, an exact right trajectory is of less emphasis but certain highlighted points are to be reached within the time. But, human trajectories could differ from demonstration to demonstration, an additional synchronisation component as used in DMP(phase estimator) has to be used to synchronise humans and robots [31]. This is where ProMP out performs DMP as it facilitates simultaneously encoding without the need for time-alignment process.

Both DMP and ProMP can encode various profiles independently. However, a relation between these profiles can be derived using ProMP which was not possible in DMP. This relation provides useful information by exploiting the mean and variance. This boundary for the mean weight vector helps in maintaining stability providing a Confidence Interval (CI). As certain parameters such as joint angles, speed are considered, ProMP can find relation between these parameters and trajectory among various demonstrations. In an attempt utilise the advantages of DMP and ProMP, Meier and Schaal proposed a method in [32] called "*Probabilistic Dynamic Movement Primitive (PDMP)*". The authors tried to improve DMP with probabilistic properties to measure likelihood that the MP is executed correctly. However, this method suffers from the absence of data-driven generalization which could result in trajectories deviating from demonstrations [33]. And because of it's unstable encoding nature (until now : anticipating future developments and improvements) and lacking data-driven generalization, PDMP is not considered for this research.

3.2.3 Hidden Markov Models (HMM)

The mathematical model developed by Baum et al. in a series of publications [34, 35, 36, 37, 38] is an probabilistic approach to determine is a state is directly visible to the observer. A mathematical representation using HMM could be made only to a finite model which can have a probability distribution over an infinite (not necessarily) number of possible sequences. A HMM model consists of state based modelling, where a number of states could be visualised as localised position on a 3-D plane. In general, a HMM algorithm could be segregated into three divisions:

- With a given HMM, the probability to generate a sequence could be determined
- Determine the optimum state sequence to generate the HMM sequence
- Identifying the necessary parameters and structures that has to be considered in a HMM for a very large data

In a HMM, a set of "*Observed Sequence*" help to predict a sequence of unknown or "*Hidden*" states. A HMM takes influence from "*Bayes Theorem*" through observations made a sequence of time steps in order to probabilistic prediction of the best sequence of hidden states [39]. Terminology related to HMM:

- Hidden States (or) States
- Observations (or) Observed Symbols
- State Transition Probabilities
  - denotes the probability of moving from a state to another state (hidden)
- Output/ Emission Probabilities
  - denotes the probability of an observing a symbol from a particular hidden state

To put it in Mathematical perspective, the probabilities of moving to state j in a timestamp $t+1$ from a state i in time t is given by:

$$a_{ij} = P(s_{t+1} = j \mid s_t = i) \tag{3.5}$$

A state transition matrix is formed for all such transitions.

$$\mathbf{A} = \{a_{ij}\} \quad \forall \ i, j = \{1, 2, ...N_s\} \tag{3.6}$$

For a given model at state i, the probabilities of observing a symbol q_k at a time stamp t,

$$b_i = P(q_k\ at\ t \mid s_t = i) \tag{3.7}$$

In terms of Observation probability matrix,

$$\mathbf{B} = \{b_i(k)\} \quad for\ \ i = \{1, 2, ...N_s\}\ \ and\ \ k = \{1, 2, ...Q\} \tag{3.8}$$

where,

- N_s is the number of states within the model
- Q denotes the number of observational symbols.

Finally, HMM is described using

$$\lambda = \{\pi, \mathbf{A}, \mathbf{B}\} \tag{3.9}$$

Where,

- π is the initial state distribution $\pi_i = P[q_1 = S_i]$
- λ is the model

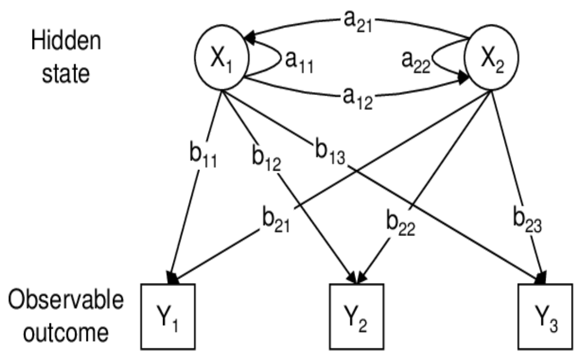


Figure 3.5: State Transition diagram of a Hidden Markov Model (HMM). [40]

A well defined and tuned HMM generally provides a stable mathematical probabilistic model. They allow online insertion and deletion of sequence profiles, making it increment/ online learning effectively possible. Because of the strong statistical foundation, effective learning could take place just my raw sequence data and any noise are better compressed than other low-level encoding techniques [41]. Just like ProMP, representation of variation in probability distributions provide a confidence interval which can be presented to the operator to provide information. In [42], a generalization of trajectories was achieved using key point identification based on significant changes in joint position and velocity. An external Dynamic Time Wrapping was used to represent the trajectories in temporal space. It also provides synchronisation of learned profiles and currently demonstrated profiles to realize incremental learning. Even though HMM out performs other low-level encoding techniques, it is computationally expensive to train. Also it is not recommended to use HMM in which time factor is involved [43] as the time spent in a given state is not explicitly captured. However, a Hidden semi-Markov model does capture time [44]. The Viterbi algorithm is cost expensive interms of memory and computation. Alternative is to use other algorithms such as forward-backward algorithm, but even this comes at a cost that doesn't under cut the former [45]. While considering HMM, smaller models are very much easier to train and as the model length increases, chances of hidden state overflow is high. Large number of hidden states increases the complexity of task generalization.

3.2.4 Task Parameterized Gaussian Mixture Models (TP-GMM)

Just like HMM, and differing from DMP and ProMP, TP-GMM uses states to derive and drive profiles. TP-GMM is an extension to the basic Gaussian Mixed Models (GMM) where several task parameters are related to the model parameters increasing the exploration capabilities. Difference lies in the fact TP-GMM adapts to different profiles automatically.

Task- Parameterised Gaussian Mixture Models (TP-GMM) was proposed by [46] to improve the generalisation or the extrapolation of typical GMM/GMR [47]. There are also other new approaches developed based on TP-GMM such as: partial observable task parameters [48], applying minimal interval control theory [49], TP-GMM to transfer skills on a soft robot [50].

In TP-GMM the task parameters ($\mathbb{TP}$) are considered as P coordinate systems, which are defined at each time step t by:

$$\{\mathbb{TP}_j : b_j, A_j\}_{j=1}^{P} \tag{3.10}$$

where,

- b_j and A_j represents the origin of the j^{th} reference frame

For a set of basis vectors $\{e_1, e_2, ...\}$, the transformation matrix A could be denoted as $A = [e_1 e_2 ...]$ respectively. All task parameters are specified beforehand. The demonstrator should have prior knowledge on the $\mathbb{TP}$ before demonstrating the task. The $\mathbb{TP}$s are usually associated with the position and orientation of the objects/ landmarks in the scene. In our case, it is the position and orientation of the virtual frames in the simulated setup.

For every demonstration $m \in M$ containing T data points of D dimensions $\{\xi\} \in \mathbb{R}^{D\times T}$ is projected in different reference frames to obtain a third order tensor dataset: $\mathbb{R}^{D\times T\times P}$. This is composed of P trajectory samples projected on P candidate frames, corresponding to matrices of D- dimensional observations at T time steps. The model parameters are defined as follows:

$$\left\{ \pi_i, \{\mu_i^{(j)}, \sum_i^{(j)}\}_{j=1}^{P} \right\}_{i=1}^{K} \tag{3.11}$$

where,

- π_i is the mixing coefficients
- $\mu_i^{(j)}$ is the mean and
- $\sum_i^{(j)}$ is the co-variance

of the i-th Gaussian component in frame j in a TP-GMM with K components.

Learning of the TP-GMM model parameters is done by maximizing the log-likelihood under the constraint that the data in the reference frames are originated from the same source, leading to the Expectation-Maximization (EM) algorithm, to update the model parameters iteratively, until the convergence is achieved [51]. With this learned model, new trajectories are reproduced. At the first step, the model retrieves GMM at each time step t. This is done by computing the product of linearly transformed Gaussian:

$$(\mu_{t,i}, \sum_{t,i}) \propto \pi_{j=1}^{P}(A_{t,j}\mu_i^{(j)} + b_{t,j}, A_{t,j} \sum_i^{(j)} A_{t,j}^{\top}) \tag{3.12}$$

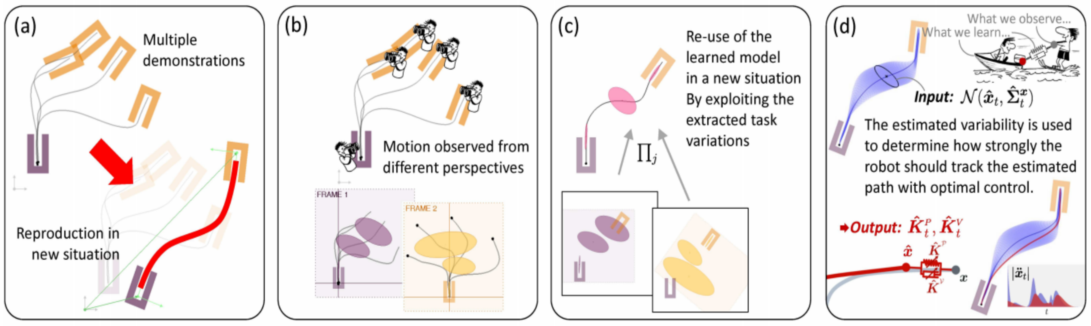

(a)
Multiple demonstrations
Reproduction in new situation
(b)
Motion observed from different perspectives
FRAME 1
FRAME 2
(c)
Re-use of the learned model in a new situation By exploiting the extracted task variations
(d)
What we observe...
What we learn...
The estimated variability is used to determine how strongly the robot should track the estimated path with optimal control.

Figure 3.6: This figure illustrates the overall approach of TP-GMM. First multiple demonstrations are performed in **(a)**. These demonstrations are observed from different reference frames **(b)** which are the task parameters. In **(c)**, the different reference frames are combined such that TP-GMM is obtained. From obtained distributions, a new trajectory is regressed in **(d)** by using a regression technique. `Image Source: [52]`

Finally, Gaussian Mixture Regression (GMR) which exploits the joint probability density function of data (both input and output) modelled by TP-GMM is used for reproduction of the new trajectory. The TP-GMM in Equation 3.12 encodes the joint distribution of the dataset. Defining the ϑ and ς to represent the sets of dimensions spanned by the input and output variables respectively, at each time step t, the datapoint ξ_t comprises of both ξ_t^ϑ and ξ_t^ς. As a result, it could be written as:

$$\xi_t = \begin{bmatrix} \xi_t^\vartheta \\ \xi_t^\varsigma \end{bmatrix}, \mu_i = \begin{bmatrix} \mu_i^\vartheta \\ \mu_i^\varsigma \end{bmatrix}, \sum_i = \begin{bmatrix} \sum_i^\vartheta & \sum_i^{\vartheta\varsigma} \\ \sum_i^{\varsigma\vartheta} & \sum_i^\vartheta \end{bmatrix} \tag{3.13}$$

At each reproduction step t, $(\xi_t^{\vartheta|\xi_t^\varsigma})$ is computed as the conditional distribution and estimated $\hat{\xi}_t^\varsigma$ is used as a command for position of the end-effector of the robot.

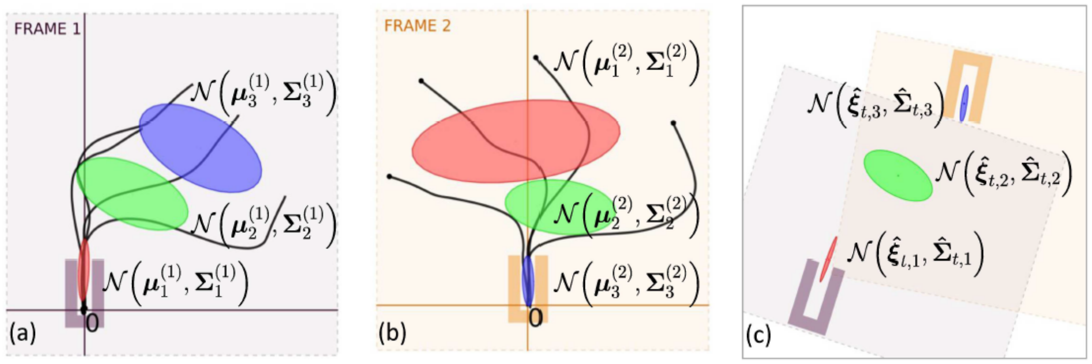

FRAME 1
$\mathcal{N}(\mu_3^{(1)}, \Sigma_3^{(1)})$
$\mathcal{N}(\mu_2^{(1)}, \Sigma_2^{(1)})$
$\mathcal{N}(\mu_1^{(1)}, \Sigma_1^{(1)})$
0
(a)
FRAME 2
$\mathcal{N}(\mu_1^{(2)}, \Sigma_1^{(2)})$
$\mathcal{N}(\mu_2^{(2)}, \Sigma_2^{(2)})$
$\mathcal{N}(\mu_3^{(2)}, \Sigma_3^{(2)})$
0
(b)
$\mathcal{N}(\hat{\xi}_{t,3}, \hat{\Sigma}_{t,3})$
$\mathcal{N}(\hat{\xi}_{t,2}, \hat{\Sigma}_{t,2})$
$\mathcal{N}(\hat{\xi}_{t,1}, \hat{\Sigma}_{t,1})$
(c)

Figure 3.7: This figure illustrates a linear product between the three Gaussian distributions from frame **(a)** with frame **(b)**, output a single task frame with the three resulting distributions in **(c)**. `Image Source: [46]`

In HMM and TP-GMM, differing from DMP and Pro-MP uses task parameters for profile generation and generalization using states of the task environment and robot. TP-GMM also uses probabilistic approach providing a confidence interval through variance to determine and provide additional information to the operator. TP-GMM could be generated only through a set of demonstrations present. The reason is, in order to find a probabilistic approach, a variance among demonstrations should be present. There is an advantage of having demonstrations, as multiple frame references could deal with noise more efficiently. Since TP-GMM could easily encode a profile based on task parameters, several parameters such as contact force, velocity could easily be modified and learned. A final step is to use a regression technique GMR to generalise and generate a trajectory. When using GMR it an additional requirement has to be met by aligning trajectories to time space. This is achieved using Dynamic Time Warping technique [22] or other similar techniques such as "*Functional Data Analysis*", in which time series are regarded as discretizations of smooth (differentiable) functions of time [53]. This approach has been successfully applied to analyze patterns and variability of speech movements in [54] and [55]. This could be adapted to dross skimming applications to analyse patterns and variability in dross skimming velocities. Since these methods takes into account the task parameters with variances, this approach is best suited for complex tasks which require tracking of specific paths and speeds.

3.3 Evaluation w.r.t Requirements

The purpose of this section is to evaluate the encoding techniques with the requirements discussed in Section 3.1. The comparison was made on two distinctive types of low-level encoding types: *Deterministic* and *Probabilistic*, *time-driven* and *state-driven*. Below is an evaluation relating to task learning and reproduction profiles.

- **Application**
  - Applicability plays a strong factor in defining how well the generated profiles are more likely to follow the correct/ expected trajectory. By taking low variance as a factor, the important constraints are determined. While considering these individual task parameters in which time is considered as an important evaluation parameter, state-driven probabilistic approaches does perform well. Therefore, a HMM or TP-GMM would be very useful.
- **Demonstration**
  - It is a known fact that, in order to obtain a confidence interval, multiple demonstrations are to be performed. Hence, probabilistic methods (Pro-MP, HMM, GMM) requires multiple demonstrations as compared to deterministic approach (DMP) which requires only one demonstration. With tasks where time factor is considered, it is obvious that more demonstrations are required to determine task parameters with low variance. Therefore, a probabilistic approach with multiple demonstrations are chosen when task parameters are aligned to time space, and a deterministic approach with fewer/ single demonstration when the task parameters put less emphasis on time space.
- **Parameters**
  - Here the possibility of determining a relation between various model parameters and determine how they could be incorporated inside the learning framework. Probabilistic approach models these task parameters into Gaussian components. Therefore, Pro-MP, HMM, TP-GMM could be used.
- **Generalisation**
  - As by the literal meaning, generalised profile could be observed only through multiple demonstrated data. Probabilistic approach which provides a variance allows a better generalisation compared to deterministic approach. Amongst probabilistic approach, TP-GMM provides more generalisation with less computation, memory cost than HMM and Pro-MP.
- **Scaling**

  - Scaling in terms of temporal behaviour does go well with time-driven low level encoders as it allows manipulation of coefficients. These coefficients could relate to various task parameters such as velocity, contact force, and other localised movements.

- **Robustness**
  - Robustness provides a metric on how the encoder responds to uncertain situation such as noise. Deterministic approach simply replicates the noise while reproducing the trajectory as neither they donot consider variance nor they take multiple demonstrations into account. Therefore, it is expected of all probabilistic encoders to have a robust learning framework than deterministic.

- **Encoding and Regression**
  - This has been described briefly under each encoding technique. A time-driven approach such as Pro-MP allows simultaneous encoding and regression, whereas DMP, HMM and TP-GMM use an external function such as DTW for simultaneous encoding and regression.

- **Incremental Learning**
  - Any learning framework should allow certain modifications to be done on it. These modifications are carried out in temporal space and online. All probabilistic methods require a demonstration data and hence it could be considered that the learning happens offline. However, an incremental learning could be done on these offline data. But while considering incremental learning through online modification, DMPs are most suitable as the framework allows online modification in temporal space.

- **Miscellaneous**
  - Factor(s) which are not a necessary metric for evaluating a low-level encoding technique but determines the ease of use are discussed here. One such factor is providing additional information to the trainer. For example, as extensively discussed, probabilistic methods provide a variance parameter as an additional information. This is used by the trainer to find the low variance task parameters to estimate a certainty in replicating the task. This allows the trainer to estimate a confidence interval. A statistical model always gives an estimate on whether more number of demonstrations are further required.

Summarising the above evaluation, DMP takes a deterministic approach as it requires fewer demonstration, possibility of incremental/ online learning, simultaneous encoding and regression without any need for external algorithms. DMP performs well when the system itself is less complex without any noise. However, as the system has complex movement profiles in the presence of noise, DMP fails to adapt and provide generalisation. The possibility of online learning could be utilized to the fullest if the task environment is a structured with simple trajectories. So, DMPs are suitable for less complex tasks in an environment with very less disturbance.

When it comes to applying robotic solutions to industries, the environment is considered as unstructured and also very complex. More complex tasks increases the scope of robot to learn a wide variety of task demonstrations. Hence there is one more reason to drop deterministic frameworks (DMP) and choose a probabilistic approach. These methods contribute to include variances of task parameters providing a robust generalisation. Therefore, Pro-MP which provides generalisation and simultaneous encoding and regression is an option. However, while considering individual independent task parameters, Pro-MP does not provide a relation, and integrating all these parameters into a single framework becomes complex. An alternative is to choose HMM, where each and individual parameter can be modelled into respective states. The problem arises where, HMM uses external algorithms for synchronisation. All these comes with a cost both interms of memory and computations. If the system is a bit more complex, there is always a high probability of more (irrelevant) hidden states if the key tasks points are not chosen properly.

Requirements	Deterministic (Time-driven)	Probabilistic (Time-driven)		Probabilistic (State-driven)
Task Requirements	DMP	Pro-MP	HMM	TP-GMM
Applicability	-	+	+	++
Demonstrations for generalisation	++	+	+	+
Task Parameters	−	-	−	++
Generalisation	−	++	++	++
Learning Requirements				
Temporal Scaling	-	+	+	++
Robustness	-	++	++	++
Simultaneous Encoding & Regression	−	++	−	+
Incremental learning	++	+	-	-
Provision of Additional Info.	−	+	+	++

Table 3.1: Summarizing LfD techniques

Therefore, to summarise a HMM model could build a strong statistical framework only if the key points are chosen properly. TP-GMM contributes to learning individual task parameters and create a model. Thus a variety of tasks dependencies can easily be encoded and a relation could be derived with less computation and memory costs. Additionally, TP-GMM provides a variance index to the trainer highlighting the important feature points. This could be used to improve generalisation. It should also be noted that HMM and TP-GMM also uses eternal algorithms for synchronisation (DTW). Keeping memory footprint in mind, additional care has to be taken while extracting only the important features of a movement primitive we could use Pro-MP, HMM since there could be state overflow condition. It is because of these reasons, TP-GMM was chosen which has an efficient memory utilization and low computation costs.

4. Experimental Setup

This chapter gives a brief explanation and the steps involved in encoding dross skimming task on the robot. First, in section 4.1, the LfD framework is discussed and how it encodes the task in general perspective. Then we narrow down to our specific task (dross skimming) and discuss how the task is demonstrated, the preparation of demonstrated dataset for learning, the algorithm used to learn the dross skimming task and obtaining a trained dataset from the model in Section 4.2. The sub-question that will be answered here is:

1. (c) ***How to build a policy driven platform for dross skimming in a simulated environment?***

4.1 The LfD Framework

As discussed in Section 1.2.3, the main principle of LfD is that the end-users can teach the robots new tasks without programming. In a traditional programming scenario, a human programmer would have to reason in advance and code a robot controller that is capable of responding to any situation the robot may face, no matter how unlikely. In contrast, LfD - PbD allows the end-user to 'program' the robot simply by showing it how to perform the task. when failures occur, the end-user needs only to provide more demonstrations, rather than calling for professional help. LfD - PbD hence seeks to endow robots with the ability to learn what it means to perform a task by generalizing from observing several demonstrations. Therefore, the key factors in a typical LfD framework boils down to: "What to imitate? and How to imitate?".

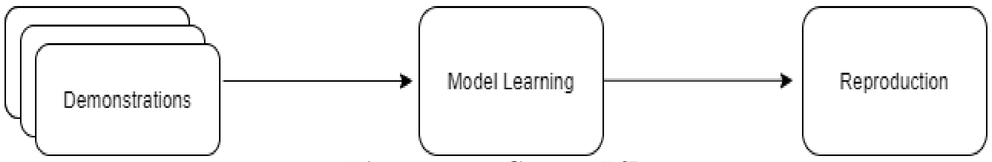


Figure 4.1: Generic LfD

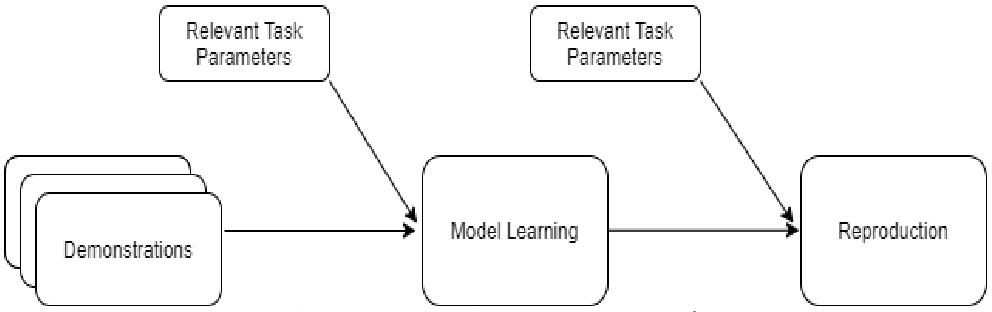


Figure 4.2: Task Parameterised LfD

4.2 Dross Skimming

With the factors discussed in previous section, we could apply an LfD framework to the dross skimming task. It is evident that, we have to demonstrate how to skim dross from the Zinc bath, also, the means by which we demonstrate is through tele-operation.

Task Demonstration

Since the assignment was aimed at generalising dross skimming task in a simulated environment, an Unity simulation set up was made which had the Zinc bath and the robotic arm with the desired end-effector. The simulation setup was then linked to a haptic device "Geomagic Touch" using the "Open Haptics plugin" in Unity from 3D-Systems. The physics on the simulation was controlled using Unity's default physics engine- "Nvidia PhysX Engine". A general Block diagram of the setup with the simulation environment is shown below.

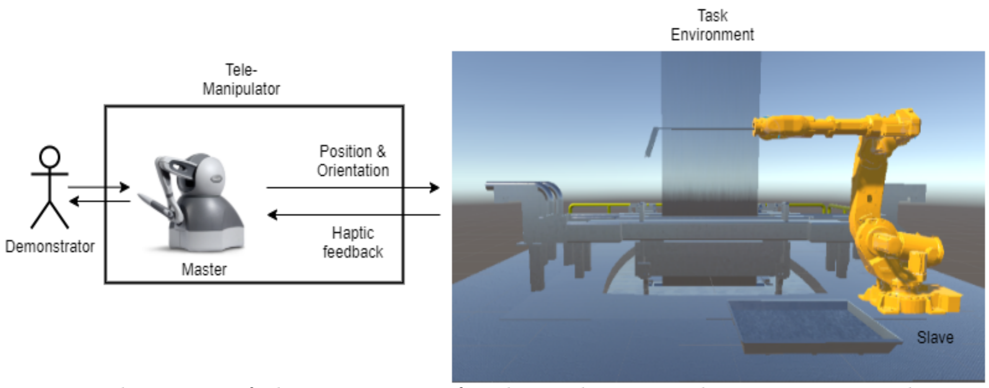


Figure 4.3: The setup of the experiment for dross skimming demonstration. There is a human operator operating the robot through a master device which controls the slave device that manipulates the environment (Zinc bath).

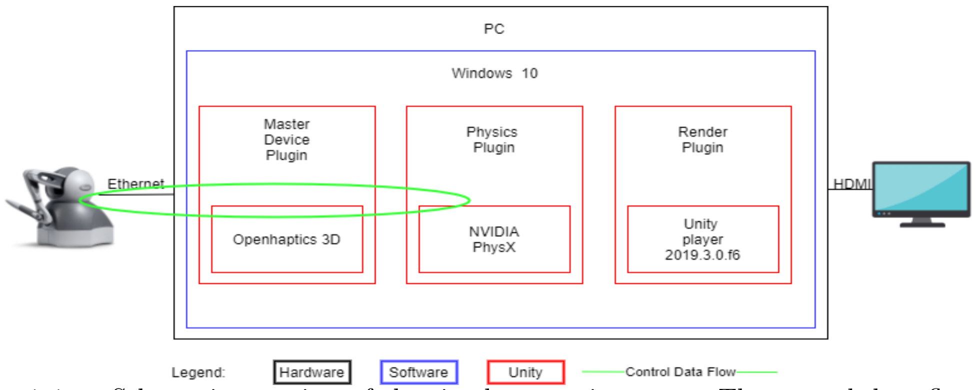


Figure 4.4: : Schematic overview of the simulator environment. The control data flow shows the interaction of the master device with the in-built Physics engine in Unity for Force feedback. `Adapted from: [56]`

A typical skimming task could be sub-divided to three tasks. Namely,

- From the initial position to one end of the bath ($\mathbb{T}_1$).
- Moving from one end of the bath to the exit point or another end. This process is also called as "Raking" ($\mathbb{T}_2$).
- After reaching the exit point, the end-effector is lifted from the bath collecting the dross. This operation is known as "Scooping". And finally, trajectory towards the dross bucket to dump the dross ($\mathbb{T}_3$).

One full dross skimming operation is learned in terms of steps or sub-trajectories in cycle as mentioned above. The reason being is that, the entire trajectory is complex and if it is demonstrated as a single trajectory, it is impossible for the model to learn and generalise. The task is demonstrated using a master- slave configuration as shown in Figure 4.3

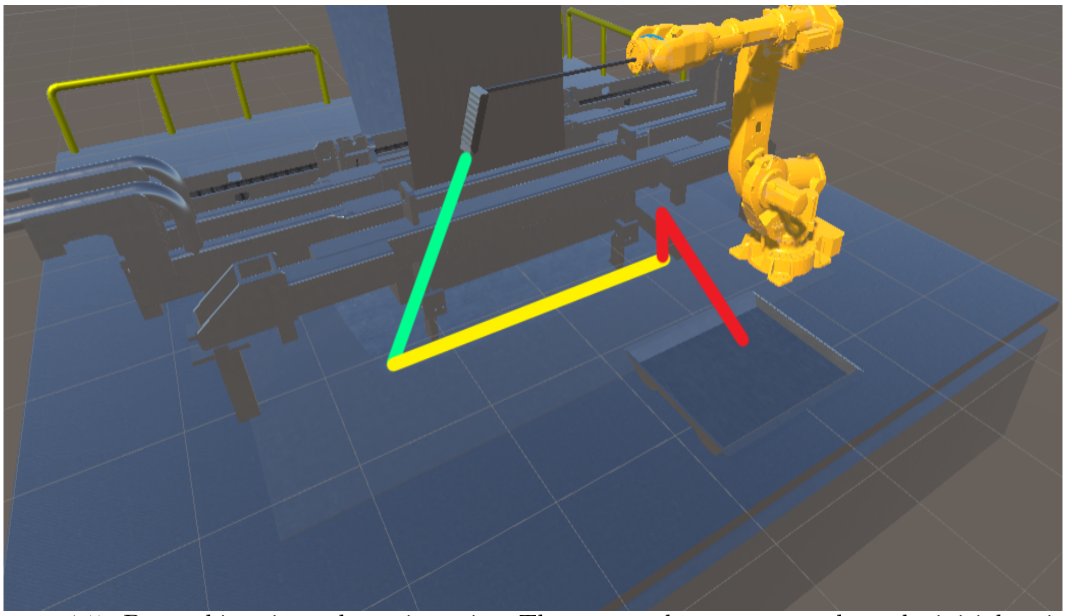


Figure 4.5: Dross skimming sub- trajectories. The green colour corresponds to the initial trajectory from Robot's initial position to the entry point in the bath ($\mathbb{T}_1$). The Yellow sub-trajectory represents the Raking operation ($\mathbb{T}_2$) and the Red colour corresponds to the Scooping operation ($\mathbb{T}_3$).

The dataset consists of the end-effector's position and orientation in 3D- plane corresponding to the world coordinates- which is the base of the robot. The data is logged at a Frequency of 100 Hz.

Task Learning

Similar to task demonstrated in previous sub-section, Task learning is also done in terms of sub-trajectories. For each sub-trajectory, a dedicated TP-GMM model is used to encode the behaviour. 3 reference frames or Task parameters are used to encode the behaviour. Over the course of learning, the reference frames are static and hence their transformations with respect to the world frame remains constant. Whereas, the end-effector frame transformations are calculated for each time step.

The main idea is to consider the objects (which modulate the movement: in this case the end-effector of the robot) in the working environment as external task parameters (or frames of references). Then, the demonstrated trajectory is projected into these frames of references and a Gaussian mixture model (GMM) is fit for each sub- task. The resulting GMMs are combined with each other to get the final TP-GMM model. In the reproduction phase, new location of the objects will be captured and used with TP-GMM to estimate a new trajectory. This implementation is based on the method proposed in [51].

LfD using Task Parameterized- Gaussian Mixture Models

Algorithm

1. **Task Demonstration**

```
for $m \leftarrow 1$ to $M$ (for each demonstration Sample of Dross- Skimming)
  – for $j \leftarrow 1$ to $P$ (for Task Parameter) record trajectory data ${}^{t}\xi^{m}$ and ${}^{t}\mathbb{TP}_{j}^{m}$ at each time step.
```

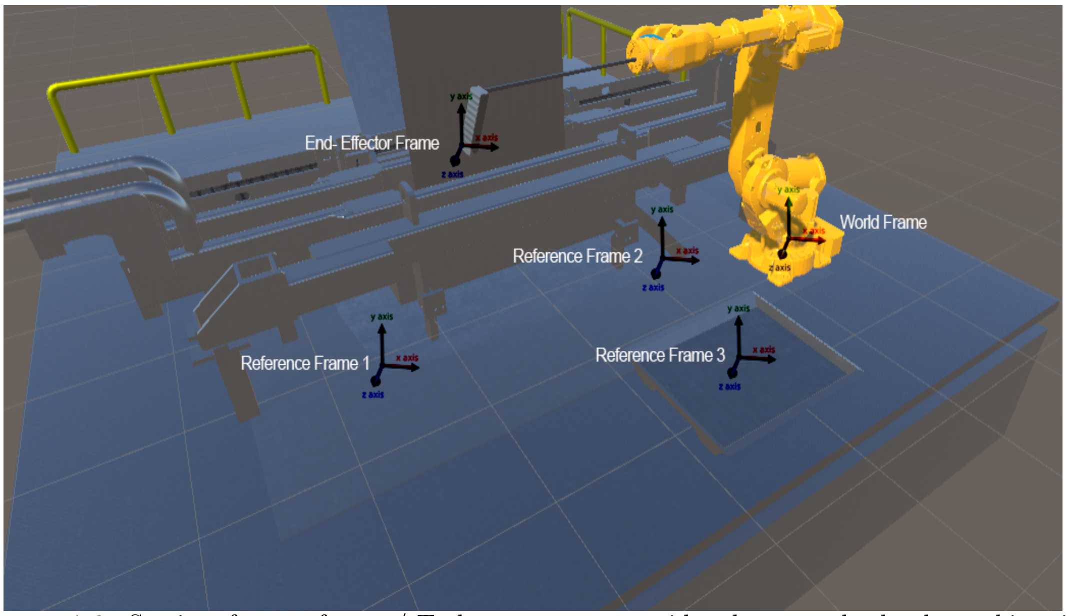


Figure 4.6: Static reference frames/ Task parameters considered to encode the dross skimming task

```
        – end
    end
```

2. **Model fitting and TP identification**

```
    for $t \leftarrow 1$ to $\mathbb{T}$ (for each task) (in this case, 1)
        – Consider all $\mathbb{TP}$s as relevant
        – Fit a TP-GMM model for the task ($\mathbb{TP} - \mathbb{GMM}_t$)
        – Identify the $\mathbb{TP}$s
        – Update $\mathbb{TP} - \mathbb{GMM}_t$ using relevant $\mathbb{TP}$
        – Store $\mathbb{TP} - \mathbb{GMM}_t$ and indexes of relevant $\mathbb{TP}$s
    end
```

3. **Reproduction**

```
        – Collect $\mathbb{TP}$s
        – Load the corresponding $\mathbb{TP}$s and $\mathbb{TP} - \mathbb{GMM}_t$
        – for $n \leftarrow 1$ to $N$ (for each reproduction step or time step)
            ∗ Use TP-GMR formulation to estimate the values of trajectory $\hat{\xi}^{s}_{n}$, at each time
              step.
        – end
```

One thing that to be noted is that, in this experiment all $\mathbb{TP}$s are considered as relevant to the experiment. That is it is assumed that the $\mathbb{TP}$s are known and available prior to the demonstrator. However, this assumption is not always a case in an unstructured environment. In such scenarios where certain $\mathbb{TP}$s are irrelevant, one has to differentiate between relevant and irrelevant $\mathbb{TP}$s based on the importance in the given task. One way to identify a relevant $\mathbb{TP}$ is to record the demonstrations including the $\mathbb{TP}$ information such as the position and orientation of the $\mathbb{TP}$, a TP-GMM model is fit to the demonstrations. Then a trial reproduction is performed where the importance of each $\mathbb{TP}$ is identified using normalized determinant of the inverse covariance matrix as mentioned in [57]. However, since we have prior knowledge on the $\mathbb{TP}$s, we go for a simpler TP-GMM implementation by assuming the $\mathbb{TP}$s to be relevant. Based on Figure 4.6, a summary of $\mathbb{TP}$s in this experiment is given below in Table 4.1. The Task parameters could also be dynamic in general (moving frames), though in this experimental setup only static $\mathbb{TP}$s are used.

Task Parameters (TPs)	Frames (`Referring to Figure` 4.6)
$\mathbb{TP}_1$	Entry point into the bath
$\mathbb{TP}_2$	Exit point from the bath
$\mathbb{TP}_3$	Dross bucket

Table 4.1: Task Parameters for the TP-GMM model

Tasks	Description	Relevant frames
$\mathbb{T}_1$	Initial trajectory from robotic arm's initial position towards the entry point of the bath	$\mathbb{TP}_1$
$\mathbb{T}_2$	Raking operation (entry point of the bath towards the exit point	$\mathbb{TP}_1$, $\mathbb{TP}_2$
$\mathbb{T}_3$	Scooping operation (exit point from the bath towards the dross bucket)	$\mathbb{TP}_2$, $\mathbb{TP}_3$

Table 4.2: A tabular description of the sub-trajectories and their relevant Task parameters

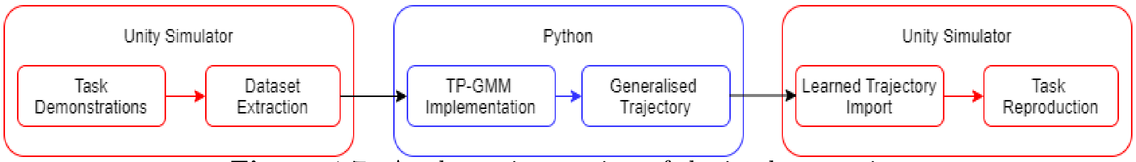
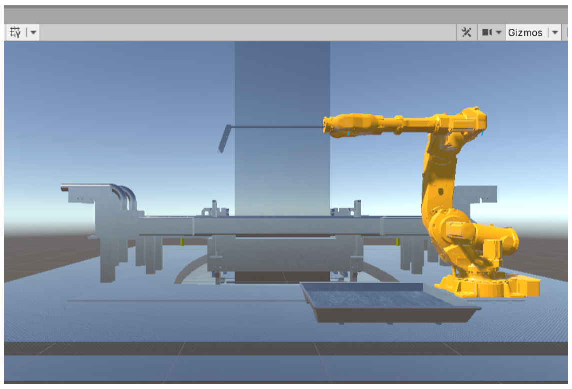


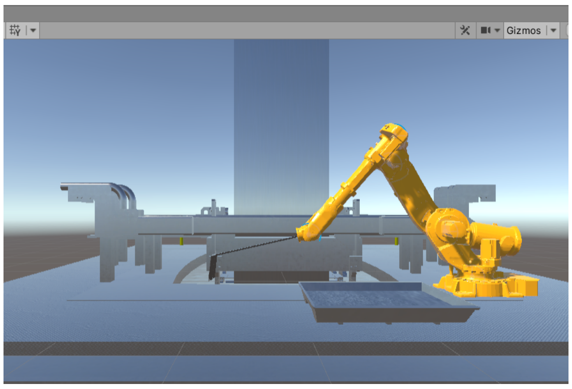
Figure 4.7: A schematic overview of the implementation

Task Reproduction

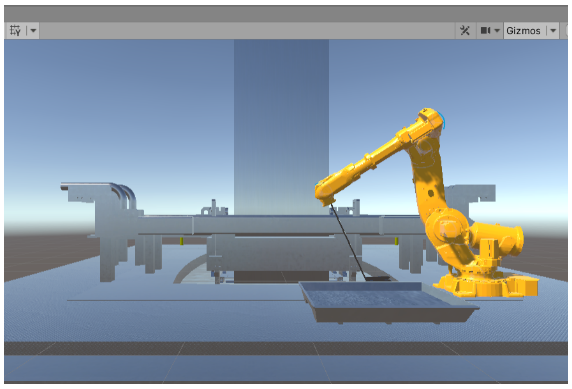
Once learning is done, the Output of the TP-GMM model is the generalised position of the end-effector over time. The output file in fed to the Unity simulator and the end-effector position and orientation are manipulated according to the model inputs. We use *FABRIK* algorithm (Forward And Backward Reaching Inverse Kinematics) [58] – a heuristic method using vectors and the concept of reaching the goal on the those vectors, to move the Robotic arm links according to the end-effector. The autonomous execution in simulation is shown below in the following Figure 4.8



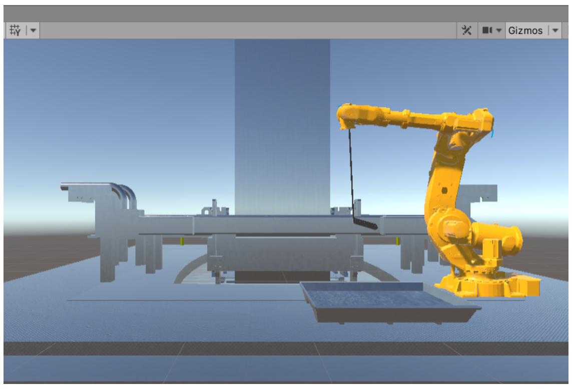
(a)



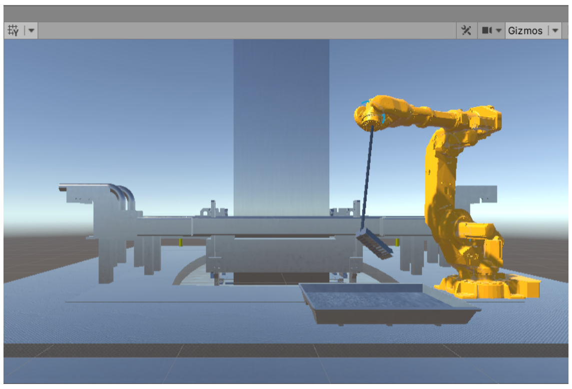
(b)



(c)



(d)



(e)

Figure 4.8: Unity: Autonomous Execution. (a) and (b) represents the sub-trajectory from the initial position of the robot to the entry point into the bath ($\mathbb{T}_1$). (c) represents the Raking operation ($\mathbb{T}_2$). (d) represents the Scooping operation. And finally, (e) represents the final sub-trajectory of exit point from the bath to the dross bucket for dross dumping ($\mathbb{T}_3$).

5. Results, Discussions and Conclusions

Now that the proposed Autonomous dross skimming is described, the next step is to validate it. This Chapter focuses on the experimental validation of the simulation proposed in Chapter 4. In this chapter, the autonomous dross skimmer is evaluated against the manual (tele-operated) dross skimming with the help of certain evaluation parameters considered in Chapter 2. However, not all evaluation parameters are considered since some parameters could not be quantified for comparison. For example, collision detection and deviation evaluation parameter cannot be quantified but it is a critical safety parameter to be taken into consideration with higher priority. Therefore, this parameter is not compared with manual dross skimming however, it is expected to be implemented in the autonomous execution to avoid all kinds of collision and to notify the user if the trajectory deviates. Other factors such as, the dimensions of the bath and the level of Zinc are considered to be constant during the course of implementation and hence they are disregarded for this experimental validation. The sub-question that will be addressed in this Chapter is:

1. (d) ***How the key performance factors could be validated with respect to autonomous dross skimming in a simulated environment?***

5.1 Results

5.1.1 Task Generalisation

One metric to analyse or evaluate the degree of learning is to compare the demonstrated trajectories and learned trajectory. In order to generalise, or to make sure that the learned trajectory does not deviate much, additional demonstrations are necessary. Deviation here denotes the variance factor. If the variance is more, the learned trajectory could deviate more.

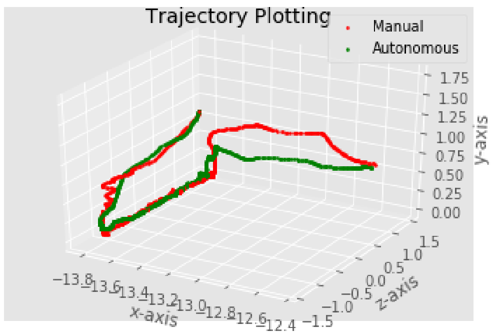


Figure 5.1: Average of all Demonstrated trajectories Vs Learned trajectory of dross skimming

5.1.2 Disturbance Induced in the Bath

To evaluate the performance is to determine the disturbances caused in the Zinc bath during the skimming operation.

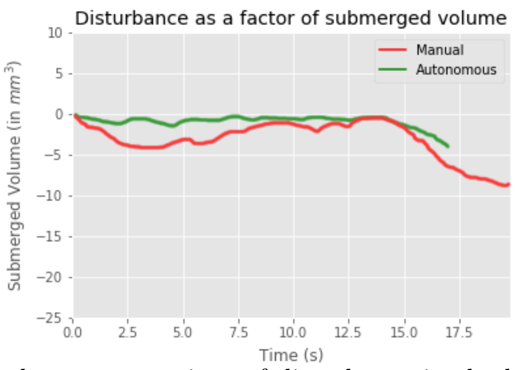


Figure 5.2: The plots show a comparison of disturbance in the bath. Average of all Manual Raking operation vs the Raking operation of Autonomous execution. Level 0 on the y-coordinate denotes the level of Zinc in the bath. The smoothness in trajectory results in creating less disturbance in the bath.

5.1.3 Velocity of Dross Skimming

To evaluate the performance index few experiments were conducted. One experiment was to compare the time taken to complete one full trajectory (time taken to complete trajectory $\mathbb{T}_1$+ $\mathbb{T}_2$+ $\mathbb{T}_3$) and individual time taken to complete each sub-trajectory ($\mathbb{T}_1$, $\mathbb{T}_2$, $\mathbb{T}_3$). For this experiment, dross factor is excluded and comparison is based only on the time factor.

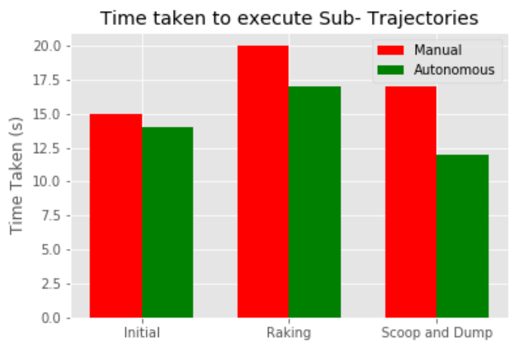


Figure 5.3: Experiment 1- Time Comparison of sub-trajectories between average of all demonstrated sub-trajectories and autonomous execution

5.1.4 Time Period for Dross Removal

In another experiment, the overall time period to remove a certain amount of dross in both autonomous and manual operation was conducted. The results of this experiment are shown below in Figure 5.4. An efficiency metric is placed based on the number of balls removed in a certain amount of time. Successful removal of each dross ball adds to 3.333% efficiency. A penalty factor is also included where in 2% of efficiency is deduced in case of undesired execution such as dross spill.

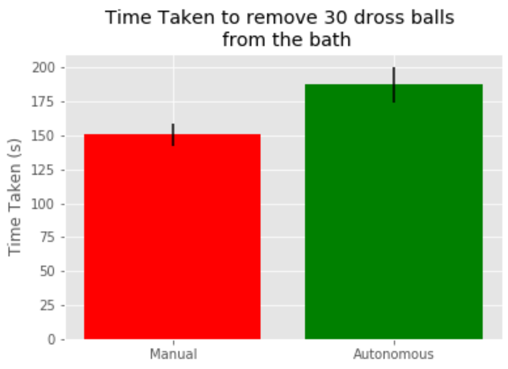


Figure 5.4: Experiment 2- Overall time taken to remove the dross in both manual execution and autonomous execution

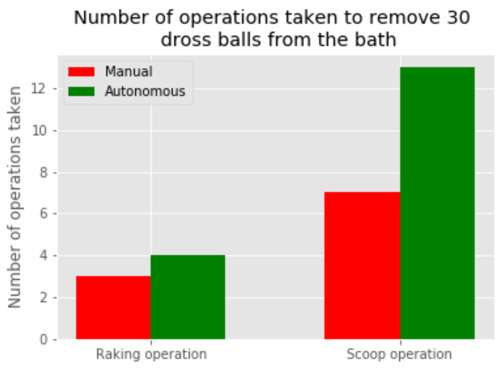


Figure 5.5: Experiment 2 - With the same setup of 30 simulated balls, this plot shows how the sub- trajectories compare against manual and autonomous dross skimming with respect to number of operations taken to achieve the particular task. Currently, in autonomous execution, the control to choose between Raking and Scooping is manually selected by the human worker during runtime.

5.2 Discussions

5.2.1 Task Generalisation

The variance factor plays a major role in terms of probabilistic modelling. If the variance of the learned trajectory is high, it is less generalised and hence therefore, the learned trajectory could deviate from the demonstrated trajectory showing stereotypical behaviours. In such cases, more demonstrations are required to bring the variance factor down. From Figure 5.1, it is seen that the learned trajectory is almost similar to that of what is being demonstrated. Except for the last part, where during the scooping and dross dumping operation, a considerable amount of variance is encountered. The reason is that, the number of demonstrations considered for generalisation was less compared to the raking operation. The reason is because, the robot is actually moving in free space, away from the bath environment, and there is very little possibility for the robot to collide with the bath. Therefore, less generalised dross dumping operation with a lot of variance doesn't affect the performance of the autonomous dross skimming and reduces the training time. Therefore, few demonstrations were needed to learn this part of the trajectory. However, for the initial part and especially during the "Raking" operation, the learned trajectory has to be as close as possible to that of the demonstrated trajectory because, the end-effector is the bath environment and any deviation could result in damages and thus, large number of demonstration were needed to generalise. The Figure 5.1 reflects the same behaviour of the learned trajectory as it is as close as possible to what is being demonstrated.

5.2.2 Disturbance Induced in the Bath

It is evident from Figure 5.2, that the disturbance which is measured as a factor of end-effector position in the y- coordinate, caused due to autonomous execution is less. this is because the raking operation in autonomous execution is almost smooth thereby causing less disturbance in the bath. It maintains the level of Zinc as a reference and performs the trajectory. Which was not possible in manual tele-operation because it is difficult to maintain a constant level while performing the trajectory. The frequency of variation between these is directly proportional to disturbance induced in the bath. A trajectory is smooth if the frequency of variations in the disturbance is less. However, this factor alone doesn't determine the disturbance as one has to keep in mind the optimum velocity with which the raking operation is performed. The lower the velocity, the less disturbance is caused.

5.2.3 Velocity Dross Removal

With the velocity set to lower optimum level during autonomous execution, from Figure 5.3 it takes less amount of time than average of all manual operation for all sub-trajectories ($\mathbb{T}_1$, $\mathbb{T}_2$, $\mathbb{T}_3$). Hence, in terms of time taken to perform one trajectory, the autonomous execution is faster (43 *Seconds*) even with the velocity set to optimum lower bound, when compared to manual execution (52 *Seconds*) to complete one trajectory.

5.2.4 Time Period of Dross Removal

Now that we have seen how the sub-trajectories compare against each other in execution, we now do another experiment, where they are put to use for a more realistic scenario. For this experiment 5.1.4, an Unity package from Nvidia called "Flex" is used to simulate the liquid dross particles. The amount of dross in the bath is equal to the number of simulated balls. A metric to evaluate the dross skimming to quantify the number of balls removed over a period of time in simulation. We try to place an evaluation criteria based on this concept in the experiment which gives an overall time taken to remove certain amount of dross from the bath. So, we simulate 30 dross balls in the bath and compare the performance in both manual and autonomous dross skimming.

From Figure 5.4, variance is expected in manual dross skimming but there is also some variance during autonomous execution. This is seen because of the velocity of the autonomous execution which could be varied during run-time by the operator. This means that the optimum speed at which the autonomous execution has to be carried out could be set by the operator. It could be inferred that the manual dross skimming out performs the autonomous execution by a considerable margin. The reason is the ease of skimming dross in manual tele-operation. The performance is improved due to the fact that, tracking dross in manual tele-operation is easy and straight forward. Half way down the experiment, it was noted that most of the dross was concentrated near the exit of the bath. Since in manual execution, the operator has the possibility to manually move the robot end-effector towards the specific target and perform the skimming operation. This improves the performance by reducing the time required to skim a given quantity of dross from the bath.

To investigate the performance deterioration in the autonomous execution, the number of operations to achieve a sub-task was counted. It is from Figure 5.5, the setup took around 3 Raking operations in manual, 4 Raking operations in autonomous execution to rake the dross from one end to the other end. The performance is on par against each other as there is no much decrease on efficiency during the raking operation. However, the manual execution could skim the dross in just 7 skimming operations whereas, the autonomous execution took 13 operations to completely remove 30 simulated dross balls from the bath. This causes a considerable drop in the efficiency of dross removal. The reason is also due to the fact of the tracking the dross and adjusting the end- effector (scoop) to remove the dross in an efficient manner.

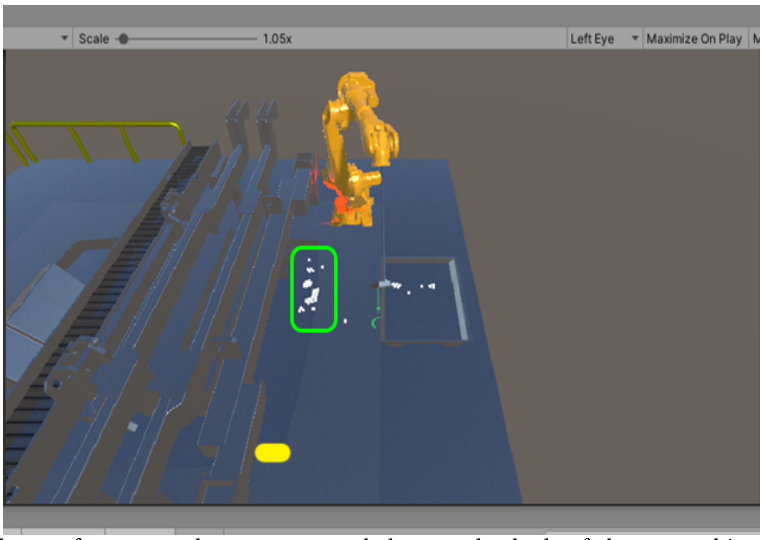


Figure 5.6: The performance decrease caused due to the lack of dross tracking in autonomous execution

The Figure 5.6 explains the drawback in autonomous execution. Most of the dross is concentrated at the exit point ($\mathbb{TP}_2$). But the generalised trajectory follows the learned order, $\mathbb{T}_1 - > \mathbb{T}_2 - > \mathbb{T}_3$ and the cycle continues. In this case, even though the dross is concentrated at reference frame $\mathbb{TP}_2$, the robot starts all the way from $\mathbb{TP}_1$, executing $\mathbb{T}_1$ first (marked with a yellow indicator in the Figure 5.6). This takes considerable amount of time and thus, increasing the overall time taken to skim the dross.

With the same setup, fixing the time constant at $100 seconds$, and 30 dross balls, the autonomous execution was able to remove 21 out of 30 dross balls giving an overall efficiency of about 68% with the penalty factor of spilling one dross ball. In manual execution, the efficiency was 100%.

While considering a real life scenario, we find that the autonomous execution to get the fundamentals right by performing a single strategy dross skimming operation. The shortcomings are: not responding and adapting to the dynamic behavior of the dross because of lack of a dross tracking mechanism, inability of the simulation setup to choose between different strategies during execution. The dross formation is not location specific and it gets spread out through the entire bath area. This experiment performs dross skimming by choosing a specific strategy at a time. If we were to use the same set-up and strategy to clean an entire bath filled with dross, this method could take a very long period to remove dross. As the raking operation $\mathbb{T}_2$ uses $\mathbb{TP}_1$ as task parameter, the dross accumulated on the bath walls in $\mathbb{TP}_1$ could not be removed. This is because, some amount of dross could drift away from the end-effector during raking and towards the bath walls making it inaccessible for the robotic arm. In such situations, different strategy has to be applied by translating the robot base and changing the orientation of the robot and performing a scoop only ($\mathbb{T}_3$) operation near $\mathbb{TP}_1$. In another situation, where the dross gets accumulated on the bath walls near $\mathbb{TP}_2$, the orientation of the scoop has to be changed so that, during scooping task ($\mathbb{T}_3$), the drifted dross is collected by the end-effector. To achieve this, different orientations of the scoop has to be learned by changing the task parameter $\mathbb{TP}_2$, where scooping operation with different orientations correspond to different strategy of autonomous scooping task. Hence, by incorporating these different strategies according to the real life scenario and behavior of the dross in the bath, an entire bath could be cleaned using the autonomous execution.

In such a situation, the task parameters associated with the model learning has to be changed relevantly so that the model could learn new movement profiles and the robot can perform a dross skimming task relevant to that situation and condition.

5.2.5 Collision Avoidance and Deviation Check

The autonomous execution is incorporated with a safety mechanism which continuously monitors for possible collisions during run-time. The autonomous execution avoids any collision with the bath environment and if in case, a collision has to happen, the autonomous execution is stopped and the control is given to the manual tele-operation to continue with the trajectory. This is done to avoid any further damage. The autonomous execution also parallely checks for deviation in the trajectory that is being executed. In such a case, the user is notified about the deviation as a warning. Few additional demonstrations could result in a more generalised movement profile which doesn't deviate from the expected trajectory.

5.3 Conclusion

To validate the learned model, we make use of the evaluation parameters discussed in Chapter 2. These are: the disturbances caused in the bath, velocity of the trajectory, time period to remove a certain amount of dross. Other additional factors include: collision avoidance and trajectory deviation check. These factors are evaluated against a base-line: Manual tele-operated dross skimming. Initial results show that, the autonomous execution takes less time to execute the sub-trajectories ($\mathbb{T}_1$, $\mathbb{T}_2$, $\mathbb{T}_3$) when compared with manual tele-operation. This indicates that the autonomous execution induces less disturbance in the Zinc bath during Raking operation. The autonomously executed trajectory is as smooth when compared to manual tele-operation.

The autonomous execution takes less time to execute the trajectories or in other words, it is faster in executing the trajectory as it's velocity is higher as it took only 43 *seconds* to execute $\mathbb{T}_1$, $\mathbb{T}_2$, $\mathbb{T}_3$. Whereas, manual tele-operation takes considerable amount to time to perform the dross skimming trajectory. To perform the same set of tasks: $\mathbb{T}_1$, $\mathbb{T}_2$, $\mathbb{T}_3$, the manual tele- operation took around 52 *seconds*. It can be inferred that the autonomous execution faster. However the manual tele- operation outperformed the autonomous execution in terms of the overall time taken to clean the bath by removing certain amount of dross. The reason is because it is easy to track dross in manual operation. Thus in spite of being faster in terms of execution, due to the lack of dross tracking mechanism, autonomous execution takes more time to clean the bath. It's efficiency stands as 68% against manual execution, which is at a 100%. The efficiency could be improved having a human to select different scooping strategies and design a dross recognition algorithm so that the robot could autonomously execute as well as track dross and adapt the strategy of dross skimming according to the dynamic behavior of dross. Despite the fact that, the autonomous execution takes more time to remove dross, it provides much operational safety and reliability by avoiding obstacles within the task environment.

5.4 Recommendations

However, there are still improvements in this field where the possibility of multiple task demonstrations technique encoding could be taken care by a single learning framework. This means, changing the position and orientation of the robot and performing different trajectories. Currently, only a single set task encoding technique that could be learned by the robot. Also, recent development in TP-GMM allows incremental/ online learning using external algorithms which allows the operator to refine the predicted trajectory during run-time. This was not considered in this research because, the learning was done offline and not within the Unity environment. There is also a possibility to create a TP-GMM model within Unity environment leading to future developments in online learning.

An important performance criteria where this implementation doesn't match up with the manual dross skimming is the efficiency as it took more time to remove dross. This was because of the lack of a visual feedback on the robot during the autonomous execution where the robot couldn't track the dross. In a more practical scenario, a strategical learning approach to perform a more extensive skimming operations along with a visual feedback to aid the robot to track and choose the best scoop position and orientation according to the dynamic behavior of the dross. Thus, yielding a fully autonomous dross skimming solution.

The visual feedback could be implemented in two ways. One of which is a camera placement on the robot which could track dross. The robot's control system could use this information as the target and navigate the end-effector of the robot. Another method is by including a HIL approach, where an expert could use the information from a camera in the task environment as a visual feedback on an interactive module. The expert could point the target on the GUI module and this information is fed to the robot and the robot's end-effector could navigate to that particular target highlighted by the operator. Ofcourse, this requires continuous monitoring from the human worker. This interactive set-up could make this robotic solution efficient and reliable, yielding a semi-autonomous solution of dross skimming.

6. Summary

Industrial robotic solutions are replacing manual/ human workers in extreme working conditions and that is a welcoming sign. The key constraint of such robotics solution is to have autonomous manipulation skills which could perform these tasks just like humans while operating in an (un)structured environment. One such application is the autonomous dross skimming in a Continuous Galvanising Line (CGL).

The core problem of removing dross from a Zinc bath (or any industrial task in general) consists of a wide variety of complex movements/ trajectories. Instead of manually hard-coding the robot, which is challenging and requires an expert every time when alterations are encountered in the task environment. An alternate approach is to conduct task demonstrations using the robot and the robot learns from these demonstrations. Therefore, this research is aimed towards robot Learning from Demonstration (LfD). A LfD framework captures the task and it's associated parameters by observing the human demonstrations. Then, the learning takes place where the skill is encoded. The last phase, the reproduction phase where the skill is reproduced back in the task environment.

The main objective of this research is to investigate what factors are involved in dross skimming, based on a tele-operation for a robotic dross skimming application and explore the possible methods to implement an automated dross skimming in a simulated environment. Therefore, the main research question is:

1. ***How the skills of an expert can be incorporated in automating dross skimming using Learning from Demonstration with a tele-operated haptic device in a simulated environment?***

This report breaks down the main research question into 4 sub categories:

1. Evaluation factors related to Zinc bath and robotic arm (Chapter 2)
2. Learning dross skimming using low-level encoding techniques (Chapter 3)
3. Experimental Setup in Chapter 4 to implement a learned model in a simulated environment to perform autonomous dross skimming
4. Experimental Validation in Chapter 5 validates the autonomous dross skimming based on the evaluation parameters discussed in Chapter 2

Chapter 2 analyses the evaluation parameters/ factors. The sub-question answered in this chapter:

1. (a) ***What are the important evaluation criteria associated with the Zinc bath and the robot aiding the skimming process?***

The evaluation factors related to both Zinc bath and Robotic arm helps in determining how to carry out dross skimming. In Human-In-the-Loop and LfD algorithm, robot tries to reproduce human-demonstrations. It is necessary for us to first know what and how to perform the dross skimming. Since these evaluation factors are industry specific and doesn't follow a standard, a visual investigation on the particular bath and type of robot used gives us an insight. Based on these factors, it is analysed to see if there could be modifications or improvisations. And finally, these evaluation factors could be used to validate the learned model.

The second component discusses technique that encodes a human behaviour in order to create a policy to generalize dross skimming. The sub-question dealt here:

1. (b) ***Which low-level encoding techniques can be utilized for learning dross skimming operation?***

Four methods are considered and analysed, viz: DMP, Pro-MP, HMM, TP-GMM. Categorised by state-driven and time-driven, deterministic and probabilistic approaches. By evaluating:

- Probabilistic methods outperform deterministic methods, as the former include variance factor which provides an insight in choosing the key factors, improve generalization, provides additional information. Additional information provides a confidence interval for certain profiles.
- Time-driven methods are better over State-driven methods given the dross skimming task. The manipulator is expected to reach specific key points in time i.e, the dross skimming velocity has to be done with optimum velocity (not too slow, not too fast).
- For less complex repetitive tasks, where task parameters are not critical, with fewer demonstrations, DMP is recommended. As extensively discussed, DMP also allows synchronisation, increment/ online learning, which gives way for an overall easy implementation.
- On the other hand, most of the industrial tasks are more complex. The robot is expected to learn a wide variety of tasks and encode in a framework. Pro-MP, HMM, TP-GMM can encode and adapt to complex trajectories. This research is focused only onto tele-operated teaching. All these methods do well with these constraints. Pro-MP, HMM which are time-based could be used in dross skimming only by properly selecting key points. TP-GMM which is state-driven does well by generalising movement trajectories by considering large spatial and temporal variability.

Thus, it is indeed possible to create a LfD framework with a tele-operated demonstration strategy and low-level encoding techniques can contribute to automating dross skimming process in a simulation. Deterministic methods(DMP) could be used for more simple task and Probabilistic methods(Pro-MP, HMM, TP-GMM) for complex tasks. Keeping memory footprint in mind, additional care has to be taken while extracting only the important features of a movement primitive we could use Pro-MP, HMM since there could be state overflow condition. It is because of these reasons, *TP-GMM* was chosen which has an efficient memory utilization, low computation costs and the possibility to relate the task parameters to model parameters.

The third component of the research question provides an overview about the implementation process that encodes the dross skimming operation in a simulated environment. The research question dealt in this Chapter:

1. (c) ***How to build a policy driven platform for dross skimming in a simulated environment?***

The experiment was setup in Unity simulation platform, where the demonstrations were carried out using a tele-operated haptic device "*Geomagic Touch*" linked with Unity. The demonstrated dataset were extracted and fed into a Python based implementation of TP-GMM model for encoding the behaviour. Three tasks ($\mathbb{T}_1$, $\mathbb{T}_2$, $\mathbb{T}_3$) and Three task parameters ($\mathbb{TP}_1$, $\mathbb{TP}_2$, $\mathbb{TP}_3$) were taken as reference frames to learn the dross skimming. The three tasks together form one complete dross skimming operation. Once learned, the obtained dataset is then fed into the same simulation setup to perform autonomous dross skimming. This answers the third component of the main research question.

The fourth and last component of the research question mainly focuses on the validation of the experiment. That is:

1. (d) ***How the key performance factors could be validated with respect to autonomous dross skimming in a simulated environment?***

The main performance factors considered were: collision avoidance, deviation detection, disturbance induced, velocity of dross skimming and time period for dross removal. Other factors that were a part of performance factors but ignored or assumed constant were the dimensions of the bath and level of Zinc in the bath. The learning metric was measured in terms of variance. For $\mathbb{T}_1$ and $\mathbb{T}_2$, the variance was very less as the trajectory has to be as close as possible to what was demonstrated. But for $\mathbb{T}_3$, the learning was done with fewer demonstrations to save the training time. This reflected in the trajectory having more variance. Since the trajectory $\mathbb{T}_3$ was moving away from the bath towards the dross bucket in free air, chances of collision were less. Thus, having a variance margin in $\mathbb{T}_3$ doesn't raise a concern. Another learning metric was collision avoidance and deviation check. The trajectory is continuously monitored in the simulation during run-time to check for possible collisions. The autonomous execution avoids any collision with the bath environment and if in case, a collision has to happen, the autonomous execution is stopped and the control is given to the manual tele-operation to continue with the trajectory. The autonomous execution parallely checks for deviation in the trajectory that is being executed. In such a case, the user is notified about the deviation as a warning. Few additional demonstrations could result in a more generalised movement profile which doesn't deviate from the expected trajectory.

The autonomous execution is compared in terms of the execution velocity. The user can vary the velocity depending on the condition and need. Lower the velocity, less jerk is caused during execution hence less disturbance in the Zinc bath. With the velocity set to the minimum bound, the tasks $\mathbb{T}_1$, $\mathbb{T}_2$ and $\mathbb{T}_3$ was executed in 43 seconds which was faster when compared to manual tele-operation which took around 52 seconds to execute the same set of tasks. Therefore even if CGL is faster (which increases the rate of dross formation), autonomous execution could adapt to this and remove the dross at a much faster rate.

However, the time taken to remove a certain amount of dross from the bath is less in manual tele-operation. Because, the autonomous execution is fundamentally carried out at operational level and tracking of dross is not possible. Thus in spite of being faster in terms of executing the trajectories, the autonomous execution takes more number of raking and scooping operation to remove a certain amount of dross from the bath. This obviously results in taking more time. It's efficiency is around 68% whereas, the manual tele-operation has an efficiency of 100% for the same experiment.

Bibliography

[1] FJ Martínez-de Pisón, F Alba-Elías, M Castejón-Limas, and JA González-Rodríguez. Improvement and optimisation of hot dip galvanising line using neural networks and genetic algorithms. *Ironmaking & steelmaking*, 33(4):344–352, 2006. 1

[2] GK Mandal, D Mandal, SK Das, R Balasubramaniam, and SP Mehrotra. Microstructural study of galvanized coatings formed in pure as well as commercial grade zinc baths. *Transactions of the Indian Institute of Metals*, 62(1):35–40, 2009. 1

[3] David J Meneice and Steven L Boston. Method and apparatus for removing bottom dross from molten zinc during galvannealing or galvanizing, October 5 1999. US Patent 5,961,285. 1

[4] Naoki Gunji and Saburo Ito. Method and apparatus for continuously hot-dip galvanizing steel strip, June 23 1981. US Patent 4,275,098. 1

[5] Bruno Maltais, Florent Gougerot, and Robert Dumont. Acs/aluminium crucible skimmer. In *TMS Annual Meeting & Exhibition*, pages 787–793. Springer, 2018. 1

[6] TB Sheridan. Teleoperation, telerobotics, and telepresence: A progress report. In *Analysis, Design and Evaluation of Man–Machine Systems 1992*, pages 1–8. Elsevier, 1993. 2

[7] Zengxi Pan, Joseph Polden, Nathan Larkin, Stephen Van Duin, and John Norrish. Recent progress on programming methods for industrial robots. In *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8. VDE, 2010. 2

[8] Valeria Villani, Fabio Pini, Francesco Leali, Cristian Secchi, and Cesare Fantuzzi. Survey on human-robot interaction for robot programming in industrial applications. *IFAC-PapersOnLine*, 51(11):66–71, 2018. 2

[9] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013. 2

[10] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010. 2

[11] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 3, 4, 12

[12] Jangwon Lee. A survey of robot learning from demonstrations for human-robot collaboration. *arXiv preprint arXiv:1710.08789*, 2017. 3

[13] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3, 10

[14] Kenneth W Krause, Donald D DeMotte, Claude A Dinsmoor, Judy A Evans, Glenn F Nowak, Gerald A Ross, Gary J Rutledge, and Charles F Slabe. Robotic system with teach pendant, May 6 2003. US Patent 6,560,513. 3

[15] Hoo Man Lee and Joong Bae Kim. A survey on robot teaching: Categorization and brief review. In *Applied Mechanics and Materials*, volume 330, pages 648–656. Trans Tech Publ, 2013. 3

[16] Kerstin Fischer, Franziska Kirstein, Lars Christian Jensen, Norbert Krüger, Kamil Kukliński, Maria Vanessa aus der Wieschen, and Thiusius Rajeeth Savarimuthu. A comparison of types of robot control for programming by demonstration. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 213–220. IEEE, 2016. 3

[17] Luka Peternel, Tadej Petrič, and Jan Babič. Robotic assembly solution by human-in-the-loop teaching method based on real-time stiffness modulation. *Autonomous Robots*, 42(1):1–17, 2018. 3

[18] Affan Pervez, Arslan Ali, Jee-Hwan Ryu, and Dongheui Lee. Novel learning from demonstration approach for repetitive teleoperation tasks. In *2017 IEEE World Haptics Conference (WHC)*, pages 60–65. IEEE, 2017. 4

[19] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016. 10

[20] Ashwin Joy. Pros and cons of reinforcement learning. `https://pythonistaplanet.com/pros-and-cons-of-reinforcement-learning/`, 2019. [Online; accessed 9-January-2020]. 10

[21] Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Using probabilistic movement primitives in robotics. *Autonomous Robots*, 42(3):529–551, 2018. 10, 14

[22] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978. 11, 19

[23] Ronny Martens and Luc Claesen. Dynamic programming optimisation for on-line signature verification. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, volume 2, pages 653–656. IEEE, 1997. 11

[24] Daniel H Grollman and Odest Chadwicke Jenkins. Sparse incremental learning for interactive robot control policy estimation. In *2008 IEEE International Conference on Robotics and Automation*, pages 3315–3320. IEEE, 2008. 12

[25] Fotios Dimeas, Filippos Fotiadis, Dimitrios Papageorgiou, Antonis Sidiropoulos, and Zoe Doulgeri. Towards progressive automation of repetitive tasks through physical human-robot interaction. In *Human Friendly Robotics*, pages 151–163. Springer, 2019. 12, 13

[26] Luka Peternel, Tadej Petrič, Erhan Oztop, and Jan Babič. Teaching robots to cooperate with humans in dynamic manipulation tasks based on multi-modal human-in-the-loop approach. *Autonomous robots*, 36(1-2):123–136, 2014. 13

[27] Fotios Dimeas and Zoe Doulgeri. Progressive automation of repetitive tasks involving both translation and rotation. In *International Conference on Robotics in Alpe-Adria Danube Region*, pages 53–62. Springer, 2018. 13

[28] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013. 13

[29] Sebastian Bitzer and Sethu Vijayakumar. Latent spaces for dynamic movement primitives. In *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pages 574–581. IEEE, 2009. 13

[30] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013. 13, 14

[31] Yanlong Huang, Leonel Rozo, João Silvério, and Darwin G Caldwell. Kernelized movement primitives. *The International Journal of Robotics Research*, 38(7):833–852, 2019. 14

[32] Franziska Meier and Stefan Schaal. A probabilistic representation for dynamic movement primitives. *arXiv preprint arXiv:1612.05932*, 2016. 15

[33] Oriane Dermy, Alexandros Paraschos, Marco Ewerton, Jan Peters, François Charpillet, and Serena Ivaldi. Prediction of intention during interaction with icub with probabilistic movement primitives. *Frontiers in Robotics and AI*, 4:45, 2017. 15

[34] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970. 15

[35] Leonard E Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1):1–8, 1972. 15

[36] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966. 15

[37] Leonard E Baum and John Alonzo Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, 1967. 15

[38] Leonard E Baum. Growth functions for trasformations on manifolds. *Pac. J. Math.*, 27(2):211–227, 1968. 15

[39] Sanjay Dorairaj. Hidden markov models simplified. `https://medium.com/@postsanjay/hidden-markov-models-simplified-c3f58728caab`, 2018. [Online; accessed 16-january-2020]. 15

[40] Charisma Choudhury, Moshe E Ben-Akiva, Tomer Toledo, Anita Rao, and Gunwoo Lee. State dependence in lane changing models, 2007. 16

[41] Joshna Rani. A look on hidden markov models. `https://www.slideshare.net/joshiblog/advantages-and-disadvantages-of-hidden-markov-model`, 2016. [Online; accessed 17-january-2020]. 16

[42] Aleksandar Vakanski, Iraj Mantegh, Andrew Irish, and Farrokh Janabi-Sharifi. Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1039–1052, 2012. 16

[43] Tshilidzi Marwala, Unathi Mahola, and Fulufhelo Vincent Nelwamondo. Hidden markov models and gaussian mixture models for bearing fault detection using fractals. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 3237–3242. IEEE, 2006. 16

[44] Shun-Zheng Yu. Hidden semi-markov models. *Artificial intelligence*, 174(2):215–243, 2010. 16

[45] Rey Rivera. Strengths and weaknesses of hidden markov models. `https://compbio.soe.ucsc.edu/html_format_papers/tr-94-24/node11.html`, 1996. [Online; accessed 17-january-2020]. 16

[46] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9(1):1–29, 2016. 17, 18

[47] Sylvain Calinon, Florent Guenter, and Aude Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(2):286–298, 2007. 17

[48] Tohid Alizadeh, Sylvain Calinon, and Darwin G Caldwell. Learning from demonstrations with partially observable task parameters. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3309–3314. IEEE, 2014. 17

[49] Sylvain Calinon, Danilo Bruno, and Darwin G Caldwell. A task-parameterized probabilistic model with minimal intervention control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3339–3344. IEEE, 2014. 17

[50] Milad S Malekzadeh, Sylvain Calinon, Danilo Bruno, and Darwin G Caldwell. A skill transfer approach for continuum robots—imitation of octopus reaching motion with the stiff-flop robot. In *2014 AAAI Fall Symposium Series*, 2014. 17

[51] Tohid Alizadeh and Batyrkhan Saduanov. Robot programming by demonstration of multiple tasks within a common environment. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 608–613. IEEE, 2017. 17, 24

[52] Sylvain Calinon. Robot learning with task-parameterized generative models. In *Robotics Research*, pages 111–126. Springer, 2018. 18

[53] Jorge C Lucero, Kevin G Munhall, Vincent L Gracco, and James O Ramsay. On the registration of time and the patterning of speech movements. *Journal of Speech, Language, and Hearing Research*, 40(5):1111–1117, 1997. 19

[54] Peter Howell, Andrew Anderson, and JC Lucero. Speech motor timing and fluency. *Speech Motor Control: New Developments in Basic and Applied Research*, page 215, 2010. 19

[55] Laura L Koenig, Jorge C Lucero, and Elizabeth Perlman. Speech production variability in fricatives of children and adults: Results of functional data analysis. *The Journal of the Acoustical Society of America*, 124(5):3158–3170, 2008. 19

[56] Jelle Hofland. Evaluating trading and sharing control for constraint motion tasks in a domestic environment using a remote controlled semi-autonomous robot. *Master Thesis Jelle Hofland 2018*, 2018. 23

[57] Tohid Alizadeh and Milad Malekzadeh. Identifying the relevant frames of reference in programming by demonstration using task-parameterized gaussian mixture regression. In *2016 IEEE/SICE International Symposium on System Integration (SII)*, pages 453–458. IEEE, 2016. 26

[58] Andreas Aristidou and Joan Lasenby. Fabrik: A fast, iterative solver for the inverse kinematics problem. *Graphical Models*, 73(5):243–260, 2011. 26