iTP-LfD: Improved Task Parametrised Learning from Demonstration for Adaptive Path Generation of Cobot

El Zaatari, S., Wang, Y., Li, W. & Peng, Y.

Original citation & hyperlink:
El Zaatari, S, Wang, Y, Li, W & Peng, Y 2021, 'iTP-LfD: Improved Task Parametrised Learning from Demonstration for Adaptive Path Generation of Cobot', Robotics and Computer-Integrated Manufacturing, vol. 69, 102109.
https://dx.doi.org/ 10.1016/j.rcim.2020.102109

DOI 10.1016/j.rcim.2020.102109
ISSN 0736-5845

Publisher: Elsevier

*i*TP-LfD: Improved Task Parametrised Learning from Demonstration for Adaptive Path Generation of Cobot

Shirine El Zaatari[1], Yuqi Wang[2], Weidong Li[1,2*], Yiqun Peng[2]

[1] Faculty of Engineering, Environment and Computing, Coventry University, UK

[2] School of Logistics Engineering, Wuhan University of Technology, China

[*] Corresponding author: weidong.li@coventry.ac.uk

Abstract

An approach of Task-Parameterised Learning from Demonstration (TP-LfD) aims at automatically adapting the movements of collaborative robots (cobots) to new settings using knowledge learnt from demonstrated paths. The approach is suitable for encoding complex relations between a cobot and its surrounding, i.e., task-relevant objects. However, further efforts are still required to enhance the intelligence and adaptability of TP-LfD for dynamic tasks. With this aim, this paper presents an improved TP-LfD (*i*TP-LfD) approach to program cobots adaptively for a variety of industrial tasks. *i*TP-LfD comprises of three main improvements over other developed TP-LfD approaches: 1) detecting generic visual features for frames of reference (frames) in demonstrations for path reproduction in new settings without using complex computer vision algorithms, 2) minimising redundant frames that belong to the same object in demonstrations using a statistical algorithm, and 3) designing a reinforcement learning algorithm to eliminate irrelevant frames. The distinguishing characteristic of the *i*TP-LfD approach is that optimal frames are identified from demonstrations by simplifying computational complexity, overcoming occlusions in new settings, and boosting the overall performance. Case studies for a variety of industrial tasks involving different objects and scenarios highlight the adaptability and robustness of the *i*TP-LfD approach.

Keywords: Learning from Demonstration, Intuitive Programming, Reinforcement Learning, Collaborative robots (Cobots)

Symbols and Abbreviations

General Symbols	
M (m as an index)	Number of demonstrations
P (i, j, k used as indices)	Number of frames
$TotPts$ (t used as an index)	Total number of path points in the task path
$b_{i,m}=\{x, y\}_{i,m}$	The x-y coordinates of the frame i in the demonstration image m with respect to a global frame of reference
$\alpha_{i,m}$	The rotation angle of the frame i in the demonstration image m with respect to a global frame of reference
Grouping redundant frames	

$d_{jk,m}$	The relative distance between the frames *j* and *k* in the demonstration *m*
$a_{jk,m}$	The relative orientation between the frame *j* and *k* in the demonstration *m*
thresh	The threshold of determining whether two frames are redundant or not
maxlead	The maximum allowed number of lead frames
Redun	*PxP* matrix where $Redun_{jk}$=1 if the frames *j* and *k* are redundant
Removing irrelevant frames	
$Relev_j$	The relevancy score of the frame *j*
iter	The iteration index of the reinforcement learning algorithm
$\Delta Relev_j$	Disruption in the relevancy score of the frame *j* in an iteration *iter*
β_j	A random number between 0 and 1 used to calculate $\Delta Relev_j$
$\Theta Relev_j$	A temporary relevancy score of the frame *j* used in an iteration *iter*
poolsize	The expected number of relevant frames for a task
id	The index of the demonstration used in iteration *iter*
cost	The cost calculated at every iteration *iter* when different frames are chosen for reproducing a path
reward	The reward calculation for every iteration used to update the value of $Relev_j$ using the obtained costs from every iteration

1. Introduction

1.1 Introduction of TP-LfD

Collaborative robots (cobots) are affordable solutions and bring various benefits to manufacturing, including facilitating mass customisation and supporting human-robot collaboration for improved working conditions to operators [1]. In particular, cobots are known to be easily integratable into manufacturing environments due to their intuitive programming user interfaces (UIs). Thus, cobots have been increasingly adopted in manufacturing companies especially small and medium-sized enterprises, in which expert programming experience might be unavailable.

Task-Parametrised Learning from Demonstration (TP-LfD) is an effective algorithm for programming cobots intuitively to act in unpredictable settings [2]. TP-LfD takes a set of demonstrations of a cobot acting in a few varied settings, and then a model is generalised over them and a new path is reproduced for a new setting. Owing to the flexibility and intuitiveness of the approach, TP-LfD has been widely used to support various engineering applications [4-6]. Main steps of TP-LfD are the following:

1. Recording demonstrations: Fig. 1 shows an illustrative example of three user-provided demonstrations for picking and placing an object with different start locations. Each demonstration consists of an image of the initial setup and a demonstration path through which the object is picked up by a cobot and is placed into a packing box.

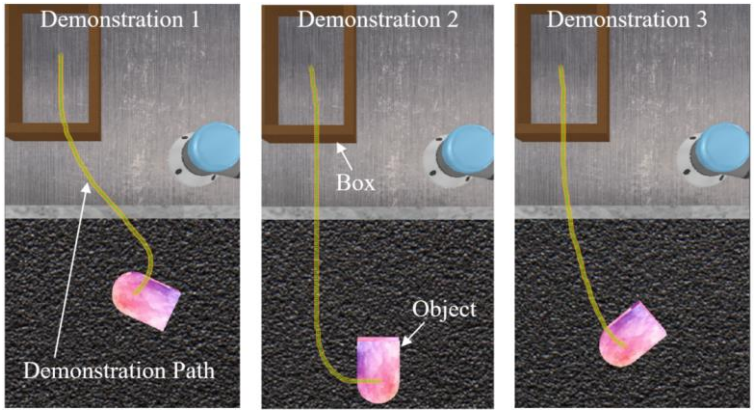


Fig. 1: Three demonstration paths for picking and placing an object in different start positions. A demonstration consists of a 2D image of an initial setup and a task path followed by a cobot to perform the task.

2. Detecting task parameters: In TP-LfD, the path of a cobot is encoded with respect to multiple frames of reference (task parameters, called "frames" in the rest of the paper). Task parameters are usually associated with locations of task-relevant objects. For example, in Fig. 2, the object and box are each associated with a frame, which is defined by a location vector and a rotation matrix. In this paper, we tackle the problem of detecting and optimising the choice of task parameters, i.e., frames.

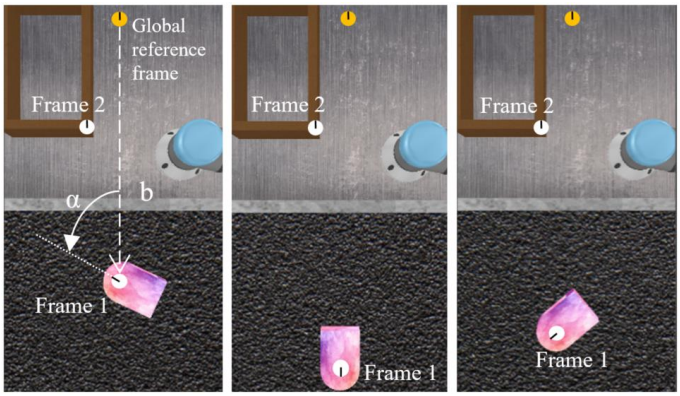
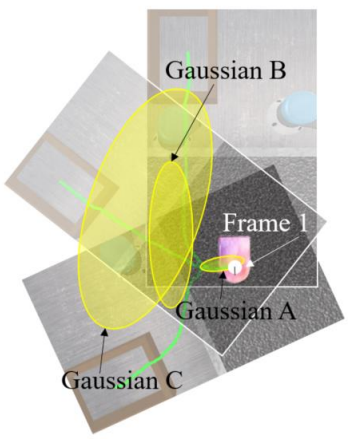
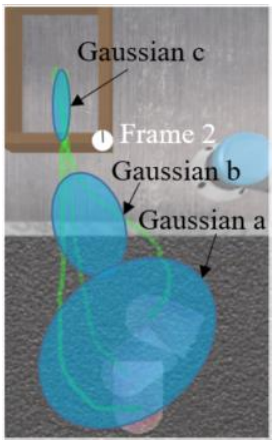


Fig. 2: Frame 1 is attached to a picked object and Frame 2 is attached to a wooden box for placing. Both frames are defined with respect to a global reference frame, by a position vector *b* and a relative orientation *α* for a rotation matrix. The location of the frame can be randomly chosen from the object.

3. Encoding the paths with respect to the task parameters: A path can be modelled as Task Parameterised Gaussian Mixture Model (TP-GMM) from the perspective of each available frame. TP-GMMs are chosen to model the paths since the central limit theorem states that normal distributions successfully models many complex systems with the least amount of prior knowledge [3]. In Fig. 3(a), Frames 1 from all the three demonstrations shown in Fig. 1 and 2 are aligned as if the paths are observed from Frame 1. Each ellipse is a Gaussian probabilistic distribution (in the paper, each ellipse is called a "Gaussian" for simplicity). In Fig. 3(a), the paths are encoded with three Gaussians (*A*, *B* and *C*) for each one third of the path. In Fig. 3(b), a similar process is carried out for Frame 2 resulting in Gaussians *a*, *b* and *c*.



(a) Paths encoded into three Gaussians after being observed from Frame 1.



(b) Paths encoded into Gaussians after being observed from Frame 2.

Fig. 3: GMMs, constituting of 3 Gaussians encoding the path as observed from Frame 1 and Frame 2.

4. Reproducing a new path: The Gaussians of the TP-GMM for each frame are regressed to obtain a Gaussian for every time step *t*. For every time step *t*, the Gaussians from all frames are multiplied together. The product factors are weighted by the inverse of the covariance of the Gaussian. The new path (called a reproduced or regression path) is sampled from the Gaussian product (Fig. 4), in a reproduction process known as Task Parametrised Gaussian Mixture Regression (TP-GMR).

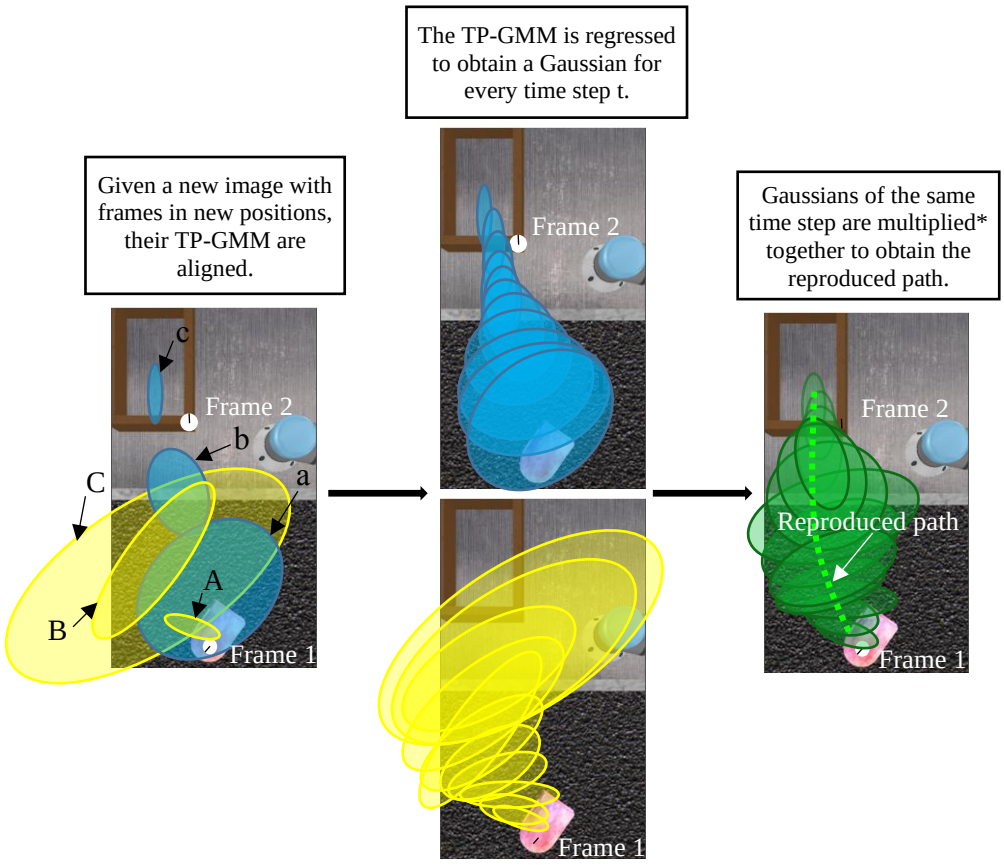


Fig. 4: The green path is reproduced in a new setting by sampling from the product of the regressed Gaussians of Frames 1 and 2.

1.2 Research challenges in TP-LfD

Frames are critical inputs for TP-LfD. Traditionally, a computer vision algorithm needs to be designed to detect these frames, in the form of objects or locations. For different tasks, the algorithm needs to be adjusted, which process is time-consuming and difficult. To facilitate the process, generic visual features can be used to represent frames.

However, many of the detected frames might be redundant or irrelevant, because the vision algorithm will detect all visual features in a working environment. For example, in Fig. 5, Frames 2 and 3 are redundant since both of them belong to the wooden box. Frame 4 can also be considered redundant since it is always at a fixed location with respect to Frames 2 and 3. Only one of them is needed for use in the successful training process of TP-LfD. Therefore, TP-LfD would eliminate all except for one. Moreover, Frames 5 and 6 are irrelevant since they belong to a task irrelevant object that is the moving conveyor belt. These frames should be eliminated since they would negatively affect the performance of TP-LfD. Removing redundant and irrelevant frames identified via the generic visual features is a tedious task, but it is crucial for improving the performance of TP-LfD.

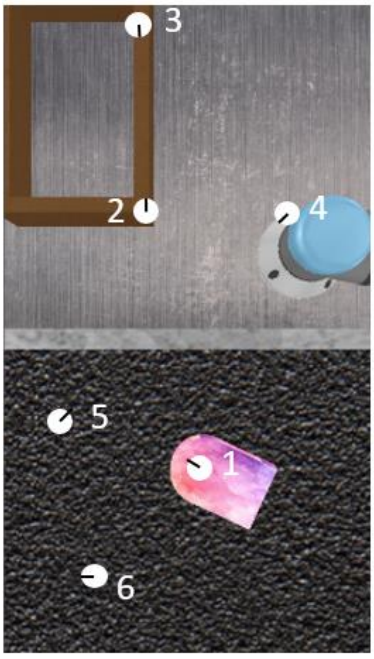


Fig. 5: Examples of redundant and irrelevant frames. Frames 2, 3, 4 are redundant since they belong to the same object/have the fixed relative position with respect to each other. Frames 5 and 6 are irrelevant since the task does not depend on them to generate the path.

To address the aforementioned challenges, in this paper, an improved TP-LfD approach (*i*TP-LfD) to support cobots in dynamic collaboration setting is presented. In the approach, a set of intelligent algorithms are designed for a variety of tasks with minimal operator interference. The details include:

(1) The *i*TP-LfD approach uses a generic visual features algorithm to optimally identify frames in demonstrations to boost the performance of the approach. In *i*TP-LfD, a statistical algorithm is

designed to group redundant frames, and a reinforcement learning based algorithm is integrated to eliminate irrelevant frames;

(2) To leverage the grouped redundant frames, an intelligent algorithm is devised to overcome partial occlusions of objects during reproductions;

(3) Based on the above intelligent algorithms, *i*TP-LfD can adapt to changes in dynamic collaborative settings in an easier, quicker and cheaper means. That is, designing task-specific algorithms or adding industrial fixtures to regulate object positions is not necessary. The performance of *i*TP-LfD was validated via multiple industrial scenarios of varying complexity, in both simulation and laboratory experiments, showing the promising potential of the approach to support industrial applications.

In the rest of this paper, further background information and literature discussions are provided in Section 2. The developed solution is explained in Section 3. The experimental setup, results and discussions are detailed in Section 4. Finally, the conclusion is drawn in Section 5.

2. Related works

In this section, some background information and discussions on related work, specifically cobot programming using TP-LfD, are given.

2.1 Programming UIs

Cobots need to be designed to closely cooperate with operators and be easily moved around shop floors for different collaborative tasks. To fulfil the tasks, cobot manufacturers designed teaching pendants that help operators program cobot easily. Based on the pendants, operators can specify way-points on the cobot's path, optimal speeds and torque parameters. However, teaching pendants are mainly used to program cobots for tasks with pre-determined settings. To be adaptive to more dynamic collaborative conditions, in recent years, researchers have resorted to creating intuitive UIs for dynamic tasks. Guerin et al. created Adjutant that allows users to define tool affordances, i.e., constraints, to be applied to different tasks [4]. Schou et al. designed Skill Based System (SBS) that allows users to program tasks using a set of parametrisable skills [5]. Each skill has a "capability" of a specific hardware component, e.g., drilling is a skill for a drill. Steinmetz et al. developed RAZOR, which is an intuitive UI that allows parametrising skills for creating programs for dynamic tasks [6]. Paxton et al. devised CoSTAR that enables users to develop programs that rely on trained computer vision models to detect specific objects dynamically. The design supports programs to be robust against changes in object positions during tasks [7]. Table 1 summarises the technical characteristics of the different user-interfaces from the above research works and the proposed approach in this paper. This paper's work stands out due to its perception capabilities that do not need adjusting/retraining for different objects, making this solution generic. Moreover, since this work relies on TP-LfD, the generated paths can have

subtle dependencies on multiple objects rather than just one. That makes it possible to be adaptive to complex tasks, including those involving HRC.

Table 1: Summary of the technical characteristics of developed approaches.

Reference	Perception capability	Support changes in object positions	Support motion/task-level programming
Guerin et al. [4]	No.	Tool constraints are interpolated to different positions. The operator moves the cobot manually to different positions.	Motion-level, i.e., it specialises in defining tool affordances and constraints.
Schou et al. [5]	Yes, but it is human-assisted, i.e., the images are transferred to the human operator for decision making.	It does not.	Task-level, i.e., given a set of pre-defined "capabilities", the developed system allows the user to easily create parametrisable skills and build task programs. A motion planner is used to create paths.
Steinmetz et al. [8]	No.	It does not.	Task-level, i.e., the user creates programs by sequencing and parametrising pre-defined skills. Kinaesthetic teaching is used to define the start and end positions of a task.
Paxton et al. [7]	Yes, it can detect markers and classify objects based on trained models.	The cobot adjusts the way-points of the path according to new positions.	Task-level, i.e., the user is allowed to create paths by specifying way-points and also build task programs using a set of pre-defined operations.
Gaspar et al. [9]	Yes, it contains object detection algorithms that adjust accuracy depending on grasping tolerance.	No, but it is easily reprogrammable to new tasks due to reconfigurable hardware.	Motion-level programming using kinaesthetic teaching, and task-level programming using UI.
*i*TP-LfD	Yes, it does not need to train models or paste markers.	The cobot automatically adjusts paths to suit new positions, based on task-parametrised learning from demonstrations.	Motion-level, i.e., the paths are generated by the algorithm intelligently.

2.2 Task parameter detection

In TP-LfD, task parameters usually refer to the positions of objects to be utilised to re-generate a new path for a new setting. These positions can be detected in a variety of ways. Classical techniques include motion capture such as in [10] and sticker markers such as in [11] [7]. Moreover, in cases where a human is involved whether as a teacher or as a collaborator, Inertial Measurement Units (IMUs) [12] or data gloves [13] are used for tracking a human's motion. These methods are not applicable in industrial scenarios due to their intrusive sense, i.e., external objects need to be attached to parts or operators. Other developed non-intrusive methods to detect human posture and movement include OpenPose [14] for full-body posture detection and Leap Motion [15] for hand motion detection. In the

research, image processing algorithms were tailored to detect specific objects. This is true when the objects are in simple colours and geometric shapes, such as those in [16] [17]. However, this might be more difficult for industrial objects with complex shapes and colours that do not stand out from the surrounding. Thus, more complex computer vision algorithms were devised and trained to detect specific objects, such as a deep learning algorithm, i.e., convolutional neural networks (CNN) in [18]. However, retraining these networks to detect new objects for new tasks is computationally expensive and requires programming expertise. Gu et al. created a system including a 3D camera and a rotating platform that were used to train a computer vision algorithm to detect the sensed objects [19]. The system recorded point clouds and saved it in a database for future detections. However, such a system includes hardware that is difficult to set up. Wang et al. developed a feature-based algorithm that used line features to learn object models to detect object pose in 3D [20]. However, their work requires the object's CAD model and is restricted to objects with line features, i.e., straight edges. Therefore, this work presents a task-parameter detection algorithm that works on generic objects and makes TP-LfD achievable with minimal sensors, i.e., a 2D camera only.

2.3 Task parameter optimisation

Another research challenge is how to optimise the selection of task parameters (frames). A few works have been done to overcome the parameter selection, such as redundant or irrelevant frames of reference. Redundant frames are defined as a set of frames belonging to the same rigid object, i.e. with fixed relative positions with each other. Due to their fixed relative position, redundant frames will have the same GMMs after training. Moreover, irrelevant frames are frames that are randomly occurring and of no relevancy to the task. Accounting for all redundant and irrelevant frames in TP-LfD degrades the algorithmic performance as the path becomes falsely biased. Ideally, there should be one frame per task-relevant object. Assuming in a large set of frames some of which are task-irrelevant or redundant, Alizadeh et al. defined an importance score $F_{t,j}$ for a Frame j at Time step (point along the demonstration path) t [21] [22]. Alizadeh and Karimi used the importance score to identify redundant frames [21]. Frames with the equal importance are deemed redundant and only one of them is accounted for during reproduction. However, this approach does not suit *i*TP-LfD for two main reasons: 1) detecting frames from visual features might lead to a large number of frames, thus generating significant computational inefficiency and difficulty to train the TP-GMM algorithm. Therefore, it is important to filter through the frames to identify redundant and irrelevant frames before executing TP-GMM; 2) in a real-life situation where data is not synthetically generated, the redundant frames will not have exactly consistent relative positions. Some of the redundant frames might be varying in position due to slight mis-localisations. Therefore, when identifying redundant frames, an error threshold needs to be introduced to account for slight mis-localisations.

On the other hand, Alizadeh and Malekzadeh used an importance score to identify irrelevant frames [22]. Frames with an importance score less than a user-specified threshold for all times t, are deemed

irrelevant and excluded from TP-LfD. However, specifying the threshold might be tricky in cases when some frames have subtle yet important effects on the path. Moreover, this method does not eliminate redundant frames in case some are missed by the algorithm in [21]. Sena et al. used the same importance score to adjust the weights of TP-LfD for different frames during reproduction [23]. This provides more accurate results at the start and end of the paths even when frame positions are changed drastically compared to demonstrations. In order to further improve the performance of this process, Huang et al. were the first to design a reinforcement learning approach to optimise task parameters [24]. Using reinforcement learning ensures that the process is improved due to the closed-loop nature of the algorithm. In their work, several frames were first initialised. Then, a reinforcement learning algorithm was used to shift the pose of these frames such that a task-specific cost function was minimised. Moreover, Huang et al. also developed an automatic frame selection algorithm [24]. Given a number of frames, their algorithm was able to identify which frames play the most important role in minimising the cost function. This helps eliminate the frames that have low influence on learning. Based on this, computation was speeded up and the performance was improved. On the other hand, there are several limits in the work of Huang et al. [24]. The approach does not tackle the question of how to visually detect frames, but rather they are specified as fixed positions with respect to the cobot's end-effector. This eliminates cases in which frames are intrinsically defined on non-static objects.

To overcome the above issues, the work presented in this paper will combine the advantages of some research and design an improved algorithm to optimise TP-LfD in terms of computation efficiency and robustness for changing industrial settings.

3. The *i*TP-LfD approach

The *i*TP-LfD approach developed in this paper consists of two general procedures: training to generate TP-GMM, and path reproduction based on TP-GMR. For the training process, there are several critical steps: 1) recording demonstrations, 2) detecting frames from the images of the recorded demonstrations, 3) grouping redundant frames, 4) generating TP-GMM, and 5) eliminating irrelevant frames. In the reproduction process, the main steps are: 1) recording a new setting, 2) detecting the relevant frames from the new image of the setting, and matching them with those in the demonstrations, and 3) regenerating a path using TP-GMR. Fig. 6 provides an overview of the procedures of training and reproduction.

In Section 3.1, an overview of the approach is presented. In Sections 3.2 - 3.6, the implementation details are discussed while the research contributions of this paper are from Sections 3.2, 3.3 and 3.5.

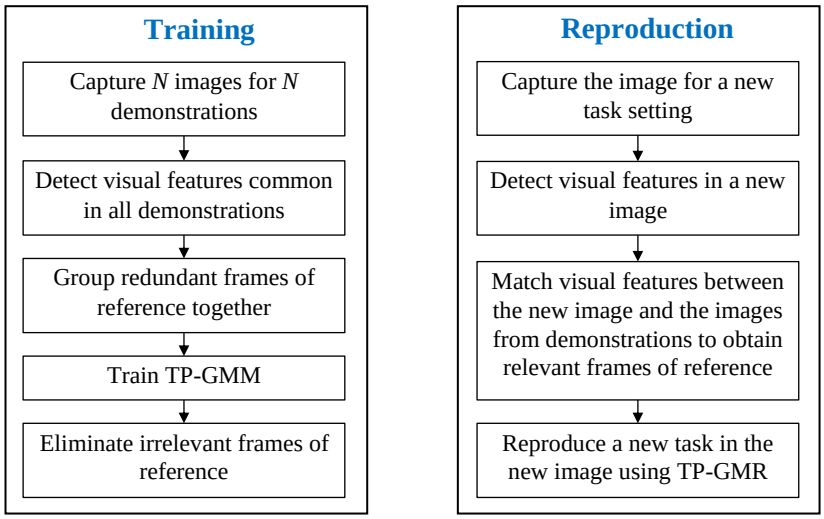


Fig. 6: The overall flow of the *i*TP-LfD approach, including training and reproduction.

3.1 Overview

In the training process, demonstrations are recorded as images. Each demonstration entails the image of a scene and a path that the cobot needs to perform in the scene. In this paper, the images in the demonstrations are recorded in 2D, i.e., top view of objects located on a surface, to simplify the computation complexity of the approach. Using the aforementioned pick-and-place demonstrations as examples, the cobot needs to pick the pink object and place it in the brown box. In each demonstration, the picked object varies its initial position while everything else is fixed and images for the processes are taken. More images would provide more demonstration variability that makes the *i*TP-LfD approach more robust. However, the more demonstrations, the lower the computational efficiency of retrieving common visual features in the images. In order to balance efficiency and accuracy, a number of experimental trials were conducted. It was observed that three to five demonstrations are enough to highlight variability, i.e., how the objects might vary positions in future task settings. After recording demonstrations, the *i*TP-LfD approach trains and reproduces the task autonomously, without requiring an operator to tweak parameters or adjust code.

In the training process, frames are extracted from the demonstration images as visual features autonomously. Redundant frames, e.g., belonging to the same object, are grouped together using a statistical method. Then, the training of the TP-GMM is executed with the frames as task parameters, resulting in TP-GMM encodings of the demonstration paths for each frame. Grouping redundant frames before generating TP-GMM is necessary since it helps decrease the total number of frames making the convergence of TP-GMM possible. Afterwards, irrelevant frames are identified to optimise the performance through a reinforcement learning process. In the reinforcement learning process, TP-GMR is used to regenerate a regression path from the trained TP-GMM.

During task reproduction in a new scenario, the goal of the *i*TP-LfD approach is to automatically generate the cobot's path based on what is learnt from the demonstrations provided previously. Firstly,

an image is taken of the new scene from the same camera position as the demonstration image. Visual features are extracted from the image and matched with the relevant frames in the demonstrations. If all the relevant frames are matched, then the path is generated using TP-GMR. If any of the relevant frames is not matched, then the algorithm proceeds by matching the redundant frames with that relevant frame. The position of the relevant frame is estimated using its relative position with respect to the redundant frame found (refer to Section 3.3 for more details). Based on the result, a regression path is generated using TP-GMR. Otherwise, if a redundant frame is also not identified, then the operation is halted and an operator interference is necessary.

3.2 Detecting visual features

Traditionally, objects are detected using sticker markers [7], pre-trained neural network models [25] or complex image processing algorithms [19]. In an effort to simplify the process of frame detection for *i*TP-LfD, in this research, visual features are used to identify frames from demonstrations. This presents a generic solution that can work for a wide range of objects without tweaking. Moreover, it requires a minimal setup, only a camera. In Fig. 7, the relevant steps are illustrated. The process is explained in more detail below.

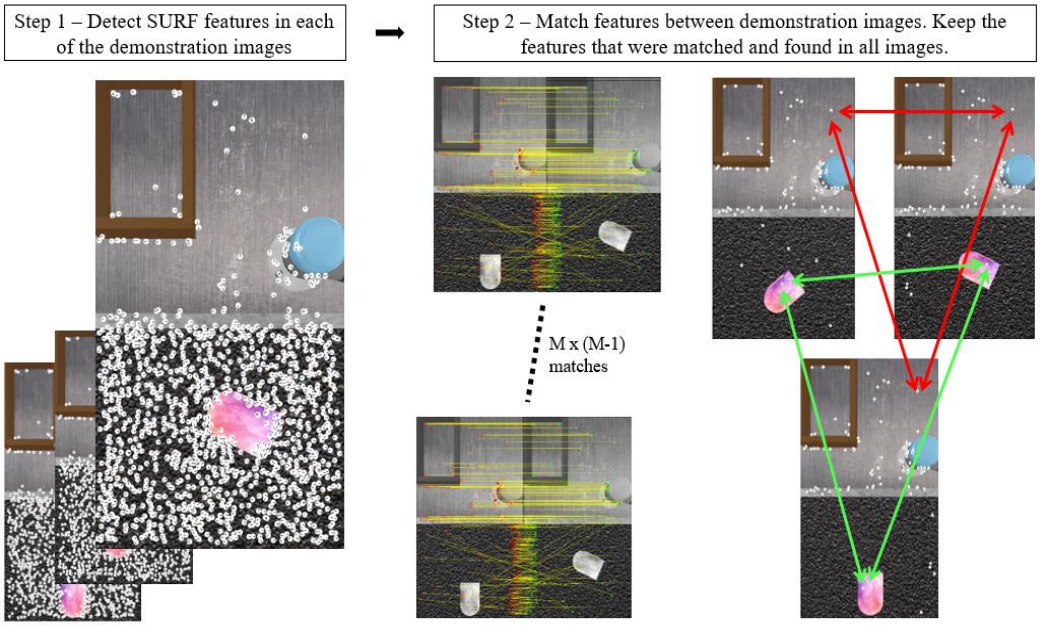


Fig. 7: The process of detecting SURF features and matching them between demonstration images.

Step 1 – Detecting visual features in each of the images of demonstrations: In this paper, Speeded Up Robust FeatuFigure 1res (SURF) are detected (oriented circles in Step 1 of Fig. 7) from the demonstration images captured by a human operator. SURF features provide a balance between detection time and number of features retrieved [26]. Moreover, SURF detects features that are scale and rotation invariant such as corners and blobs [27]. This is suitable for this paper's application in which objects might vary orientation, though not necessarily vary in size with respect to the camera.

Step 2 – Matching features between the images of demonstrations: The features from each two images are compared and identical features are associated together, i.e., matched. The matching is carried out by an exhaustive nearest neighbour search method [28]. Features that are matched across all the images of demonstrations are kept. If a feature is not found in at least one of the images, it is eliminated since TP-GMM can only be trained if a feature is found in all demonstration images. In Step 2 of Fig. 7, it can be observed that the matched features are less than the total SURF features in Step 1. Each matched feature is called a frame (e,g., the frame *i*), which is defined by a pixel position vector $b_{i,m}=\{x,y\}_{i,m}$ and an orientation $\alpha_{i,m}$, where $m \in M$, and M is the total number of the demonstration images.

3.3 Grouping redundant frames

In each demonstration, the relative position is used to identify redundant frames for grouping. The distance d and the relative orientation α between two frames, e.g., the frame j and the frame k in a demonstration m, are calculated using the following Equations (1) and (2) respectively. If the standard deviation σ divided by the mean μ is below a certain threshold *thresh* (see Equation 3), the frames are deemed redundant. A P x P redundancy matrix *Redun* is defined to represent the redundancy relationship between frames, where P is the total number of frames. In the matrix, if $Redun_{jk} = 1$, the frames j and k are considered redundant (see Equation 4).

$$d_{jk} = \{d_{jk,m} \; \forall \; 1 \leq m \leq M \; such \; that \; d_{jk,m} = \sqrt{(x_{j,m} - x_{k,m})^2 + (y_{j,m} - y_{k,m})^2} \tag{1}$$

$$a_{jk} = \{a_{jk,m} \; \forall \; 1 \leq m \leq M \; such \; that \; a_{jk,m} = \; (\alpha_{j,m} - \alpha_{k,m}) \tag{2}$$

$$if \; \frac{\sigma(d_{jk})}{\mu(d_{jk})} < thresh \; \cap \; \frac{\sigma(a_{jk})}{\mu(a_{jk})} < thresh \tag{3}$$

$$then \; Redun_{jk} = Redun_{kj} = 1 \tag{4}$$

After completing *Redun*, redundant frames are gathered in groups called objects. Lead frames are chosen such that each object has one lead frame. The rest of the frames belonging to that object are stored, to be used in case of partial occlusion. The threshold is adjusted so that the resultant number of lead frames is below a particular number, which ensures that the *i*TP-LfD approach will converge. Fig. 8 and 9 summarise the relevant steps of this process based on a statistical algorithm. Fig. 10 shows an example.

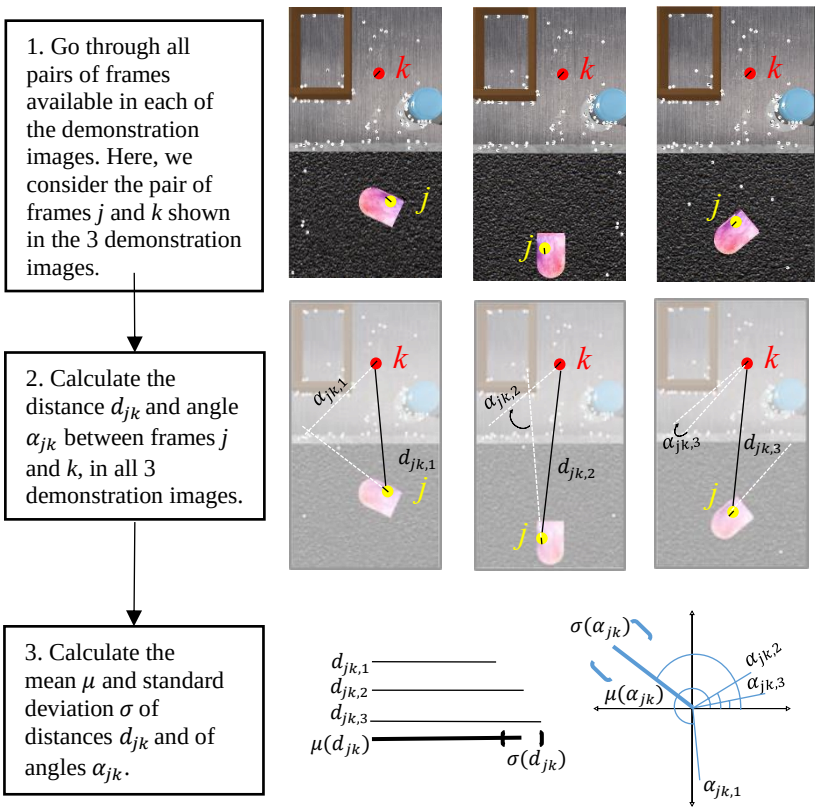


Fig. 8: Calculation of the mean and standard deviation of distances and angles between every pair of frames, as part of grouping redundant frames together.

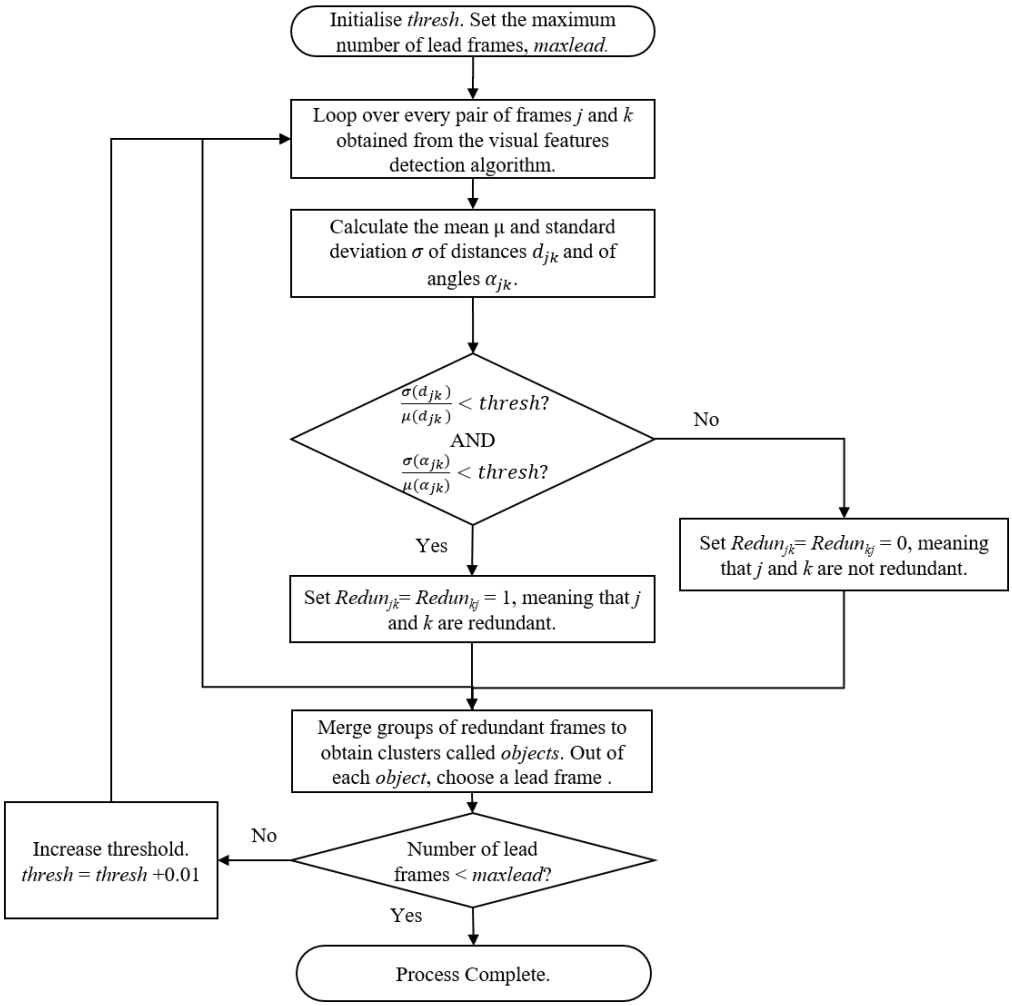


Fig. 9: Flowchart of the statistical algorithm showing how redundant frames are grouped. The threshold is increased until the total number of lead frames is less than 25.

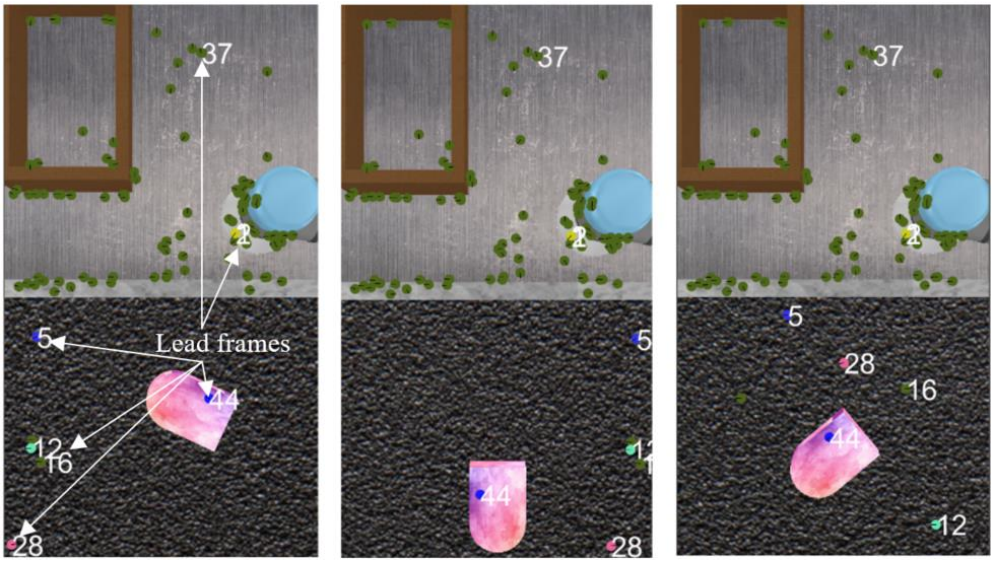


Fig. 10: Redundant frames form *objects*, i.e. the dots with the same colour. The chosen lead frame out of each *objects* is marked in white and a number.

3.4 Training TP-GMM

TP-GMM is then trained using the collected demonstration data, consisting of demonstration paths and task parameters (frames). The input to the algorithm consists of two elements, i.e., 1) a path, drawn by the user on the demonstration image, which the cobot should follow to execute the task. There is a path for each demonstration provided. Each path consists of a *TotPts* number of points defined by (x_k, y_k) pixel coordinates on the image; 2) a set of task parameters, for each demonstration image, which are the lead frames identified in Section 3.3. Each frame such as the frame j is defined by a pixel position vector $b_{i,m}=\{x,y\}_{i,m}$ and an orientation $\alpha_{i,m}$, where $m \in M$, and M is the total number of the demonstration images. In TP-GMM, the paths are encoded as TP-GMMs with respect to each of the lead frames [2]. Therefore, the outputs of the TP-GMM are a set of arrays in the length *TotPts* comprising of Gaussian distributions. Each lead frame has its own array. Each Gaussian distribution comprises of a mean vector μ and a covariance matrix $\Sigma_{t,j}$, modelling the distribution across demonstrations of a point $t \in [1, TotPts]$ along the path observed from the frame j. In a new task setting, the reproduced path will be sampled from the product of these Gaussians using a process called TP-GMR (Refer to Section 3.6).

3.5 Identifying relevancy of frames

The performance of the *i*TP-LfD approach is highly dependent on the set of frames given. If any of the frames are task-irrelevant, the performance of *i*TP-LfD will deteriorate. Therefore, a reinforced learning-based algorithm is designed to enhance the *i*TP-LfD approach by scoring the relevancy of the frames and eliminating irrelevant frames. Some critical steps of the algorithm are described below.

A relevancy score for each frame $Relev_j$, e.g., the frame j, is defined. Relevancies are initialised with equal values such that their sum equals 1.

$$Relev_j = \frac{1}{P} \tag{5}$$

where P is the total number of (lead) frames.

The *i*TP-LfD approach takes a number of iterations of computation for optimisation. In each iteration *iter*, a disturbance $\Delta Relev_j$ to the relevancy score is introduced as part of the exploration tactic in reinforcement learning. The temporary relevancy score resulting from the disturbance, $\Theta Relev_j$, is created to be used in *iter* to filter the frames:

$$\Delta Relev_j = \left(\beta_j - Relev_j\right) \times scale \tag{6}$$

$$\Theta \mathrm{R}elev_j = \left(Relev_j + \Delta \mathrm{R}elev_j\right) / \sum_j \left(Relev_j + \Delta \mathrm{R}elev_j\right) \tag{7}$$

β_j is a random number between 0 and 1 and the *scale* is usually chosen as 0.05. $\Theta Relev_j$ is a temporary relevancy score used in an iteration. The equations are designed such that it is certain that the resulting $\Theta Relev_j$ will also add up to 1 while each $\Theta Relev_j$ being between 0 and 1.

Frames are sorted in decreasing order of $\Theta Relev_j$. Given a specific *poolsize* number, the first *poolsize* number of frames are chosen. That is, if *poolsize* = 2, the first 2 frames with the maximum $\Theta Relev_j$ are chosen. Then, the TP-GMR process (Refer to Section 3.6) is executed given the chosen frames with

maximum relevancy and the path is generated. A cost function is defined to be the average of the distance between the points on the reproduced path and those on the demonstration path (Equation 8).

$$cost = \frac{1}{TotPts} \sum_{k=1}^{TotPts} \sqrt{(x_k - X_k)^2 + (y_k - Y_k)^2} \quad (8)$$

where *TotPts* is the total number of points on the path, and (x_k, y_k) and (X_k, Y_k) are the coordinates of the k^{th} point on the reproduced and demonstration paths. The cost for every *period* run is normalised to fit into the range of 0 and 1. For every *period* number of iterations, the reward is calculated below:

$$reward_{iter} = sigmoid(-5 \times cost_{iter}). \quad (9)$$

$Relev_j$ is then update by adding the $\Delta Relev_j$ of each iteration *iter* multiplied by the reward of that iteration. The algorithm converges such that the *Relev* values of frames belonging to relevant objects surpasses those belonging to irrelevant objects. In other words, the frames of maximum *Relev* chosen for TP-GMR are relevant and hence result in the best performance. The trained algorithm is able to identify the relevant frames. Table 2 summarises the algorithm to eliminate irrelevant frames.

Table 2: Pseudo code for filtering out irrelevant frames.

Algorithm Filtering Out Irrelevant Frames

Inputs:
poolsize: expected number of relevant frames
Demonstration data: images, paths, frames, and GMMs
Initialisation:
Let *period* = number of frames *P*
Let *Total number of iterations* = *period* x 10 x *poolsize*
Let *Scale* = 5e-2
Let *Relev* = 1xP vector of 1/*P*
Main Program:
FOR *iter* from 1 till *Total number of iterations*
  Reproduce:
  Set *ΔRelev* to a random 1x*P* vector of values between 0 and 1-*Relev*
  Let $\Theta Relev^{iter}$ = *Relev* + *ΔRelev*
  Sort $\Theta Relev^{iter}$ in decreasing order
  id is the index of a randomly chosen demonstration
  filtered_frames is the first *poolsize* number of frames from demonstration *id* of the highest *ΘRelev*
  Reproduce path using GMMs of *filtered_frames* only
  $cost^{iter}$ is the distance between reproduced path and demonstration path of *id*
  Update:
  IF *iter* is a multiple of *period* THEN
    Normalise costs from all previous iterations between 0 and 1
    FOR each *iter* in the past *period* iterations
      $reward^{iter}$ = sigmoid(-5 x $cost^{iter}$)
      $updateRelev^{iter}$ = *reward x* $\Delta Relev^{iter}$
      Relev = *Relev* + $updateRelev^{iter}$ */period*
    END FOR
  END IF
END FOR
Outputs:
Relevant frames = the first *poolsize* frames of the highest *Relev*

Fig. 11 displays an example for the above processes. One of the relevant frames chosen (white dot) belongs to the pink object to be picked. The other belongs to the fixed fixture of the cobot, which is at a fixed position with respect to the brown box. Hence, the second frame belong to the location that the object is placed in. The white path is generated using TP-GMR, with Gaussians of the relevant frames only. More details about the performance of the algorithm in comparison to the ground truth and other solutions are included in Section 4.

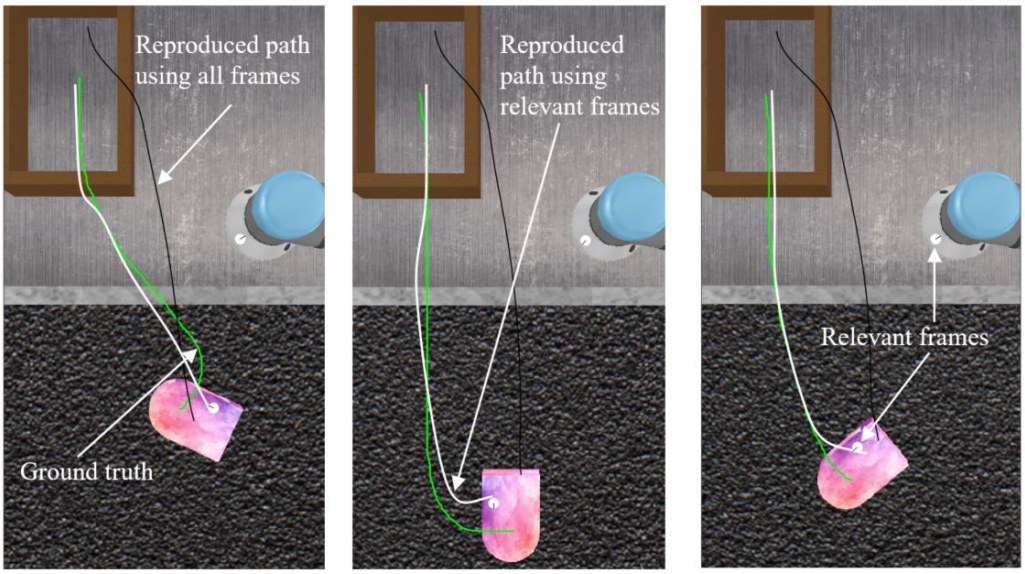


Fig. 11: The training results of the reinforcement learning based algorithm on the demonstration images. The white path is generated using the TP-GMR process accounting only for the relevant frames. The white frames are the ones that the algorithm identified as relevant.

3.6 Reproducing the task in new setting (TP-GMR)

After the training process taking place off-line as explained in Sections 3.2 to 3.5, the cobot needs to apply TP-GMR on-line for new settings (a reproduction process). Firstly, an image is captured in the new setting, i.e., similar images to those in demonstrations but with objects varying positions. In order to reproduce the path using TP-GMR, the relevant frames according to the procedure presented in Section 3.5 should be identified in the new setting. Therefore, the second step is to match the visual features, i.e., frames, in the new image with the relevant ones in the demonstration image. If all the relevant features are matched, the path can be reproduced using the TP-GMR process. The inputs for TP-GMR include the following: 1) The relevant frames detected in the new image. Each frame (denoted the frame j) is defined by a pixel position vector $b_i=\{x,y\}_i$ and an orientation α_i; 2) The GMMs for each relevant frame obtained from the off-line training of TP-GMM. Each frame has an array of the length *TotPts* of Gaussian distributions. Each Gaussian distribution comprises of a mean vector μ and a covariance matrix $\Sigma_{t,j}$, modelling the distribution across the demonstrations of a point $t \in [1, TotPts]$ along the path observed from the frame j. In TP-GMR, the GMMs are multiplied together and the reproduced path is sampled from the product, adjusted for the new positions of the frames (Fig. 3).

More details on the algorithm refer to [2]. Fig. 12 shows an example of reproducing the task in a new setting, after the two relevant features have been located in the image.

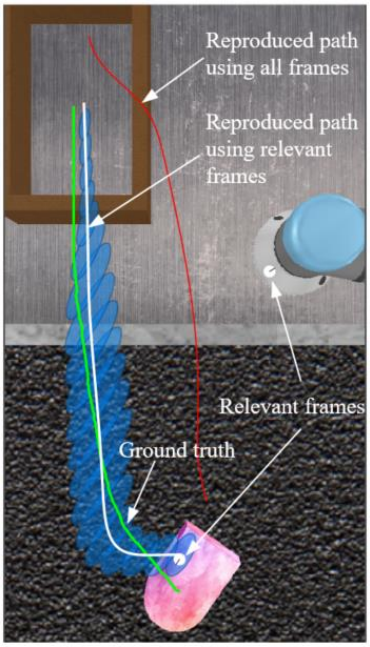


Fig. 12: An image of a new setting in which the pink object is in a new position, unseen in the demonstrations. The relevant frames of reference are identified in the new image automatically. The path is then generated using the TP-GMR process using the GMMs of the relevant frames.

If any of the relevant features could not be matched, possibly due to it being occluded in the new setting, then the *i*TP-LfD approach attempts to match any of the redundant frames. The redundant frames are identified in the off-line training as described in Section 3.3. If the matching is successful, then a position transformation is done to estimate the relevant frame's position from its detected redundant frame. Redundant frames have a fixed relative position with respect to each other. Finally, once the whole set of relevant frames are identified in the new image, the path is reproduced. Fig. 13 further provides explanation and an example of reproducing the task with occluded relevant frames.

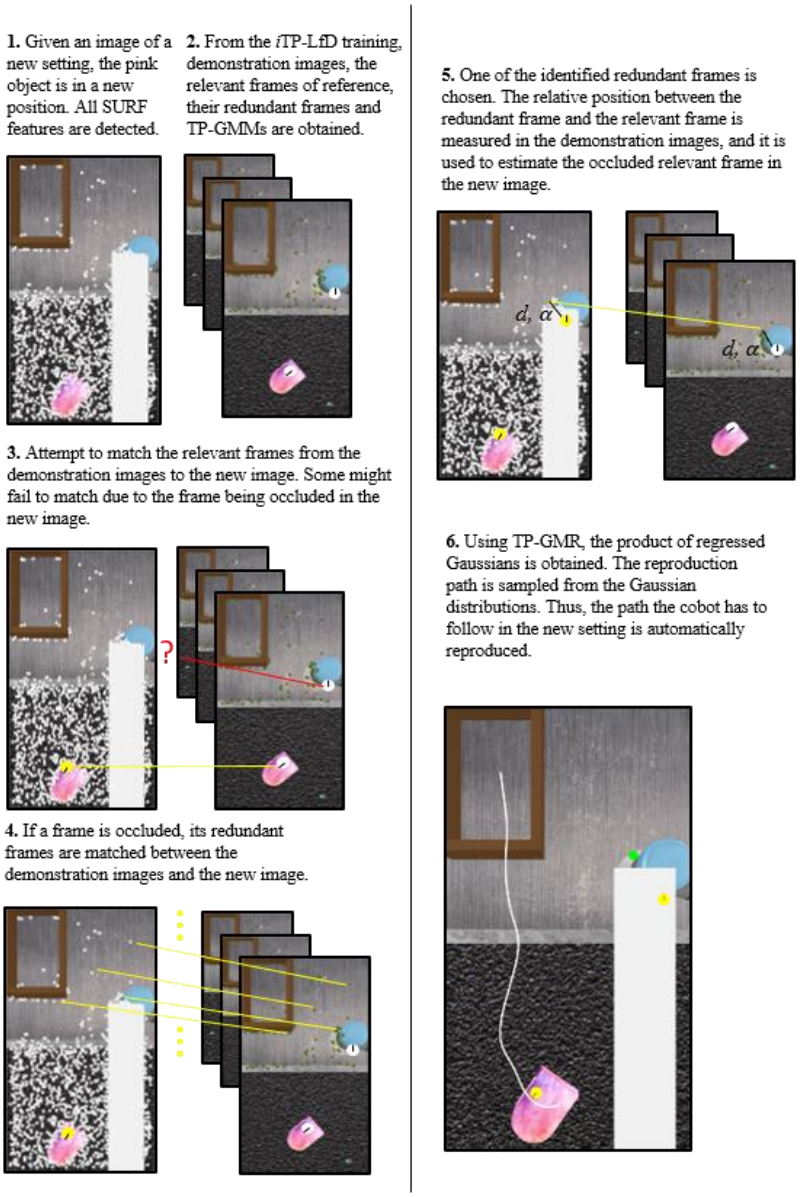


Fig. 13: A new image in which an unusual obstacle hides one of the frames. *i*TP-LfD matches a redundant frame and estimates the position of the hidden frame. Then, it reproduces the path using TP-GMR.

4. Experimental Results and Discussions

4.1 Effect of the varying *poolsize* parameter

The aim of this subsection is to investigate how the performance of the algorithm is affected by varying the *poolsize* parameter. When identifying the relevant frames, the *i*TP-LfD approach requires the user to input the expected number of relevant frames, called the *poolsize*. Since the *poolsize* is so far a user-specified parameter, the possibility of automatically identifying the correct *poolsize* is explored. Moreover, the effect of misstating it by the user is investigated, i.e. what would happen if the user provides a false *poolsize*. To fulfil the purpose of this investigation, *i*TP-LfD was executed on synthetic data while varying the *poolsize*. The synthetic data is such 20 frames are scattered in random positions and orientations. All the frames are independent of each other, i.e. there are no redundant

frames. Two of these frames are task-relevant, where the task is to go from one frame to the other; hence, the correct *poolsize* is 2. The training set is three demonstrations and the results are shown on a new setting. Table 3 and Fig. 14 provide some results.

Table 3: The performance of each algorithm with respect to the ground truth.

Algorithm	**Distance from ground truth**
Alizadeh and Malekzadeh's approach [22]	0.1988
*i*TP-LfD (*Poolsize* = 1)	0.4245
*i*TP-LfD (*Poolsize* = 2)	0.1988
*i*TP-LfD (*Poolsize* = 3)	0.2718
*i*TP-LfD (*Poolsize* = 7)	0.3874

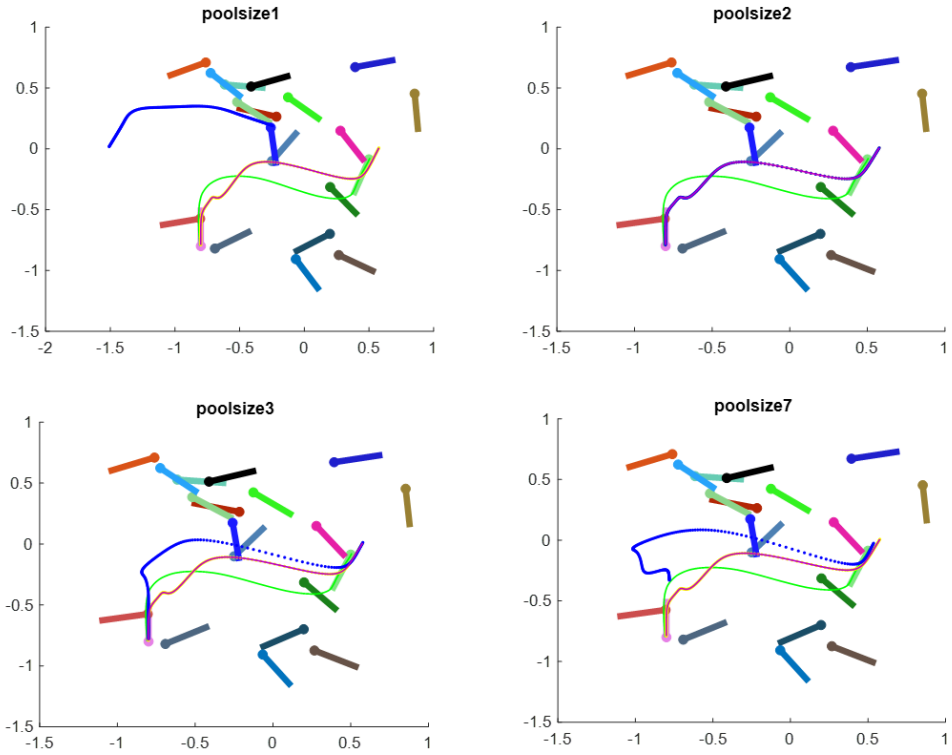


Fig. 14: The results on the validation demonstration: ground truth (green), Alizadeh and Malekzadeh's approach (purple), and the generated path using *i*TP-LfD with variable *poolsize* (blue).

The results show that choosing the correct *poolsize* is imperative to reaching optimal results. However, it also shows that one additional frame would still lead to meaningful results. For example, Fig. 14 shows that the path did not significantly change when the *poolsize* was varied from 2 to 3. However, when the *poolsize* was increased to 7, the path detached at the start which led to dysfunctional results. Moreover, the results show the possibility of automatically finding the optimal number of frames by minimising the difference between the generated path and the ground truth.

4.2 Effect of number of lead frames on the reinforcement learning algorithm

As previously mentioned, the TP-GMM algorithm does not converge if given a high number of frames (e.g., more than 50). That is why it is important to eliminate some of the redundant frames before training the TP-GMM. However, the performance of the reinforcement learning based algorithm was also found to be sensitive depending on the number of frames inputted. In this sub-section, the performance of the reinforcement learning algorithm is investigated as the number of lead frames are varied. The purpose of this investigation is to determine the maximum number of frames that yields an acceptable performance, and then tuning the redundancy algorithm accordingly.

The experiment was run on the synthetic data in Section 4.1. The number of frames was varied in increments of 5 from 5 to 30. Moreover, none of the frames were redundant and two of them where relevant. The task was to go from the first relevant frame to the second, as they varied positions. The reinforcement learning algorithm was responsible for identifying these two relevant frames. The algorithm was run 10 times to obtain the success rate shown in Fig. 15.

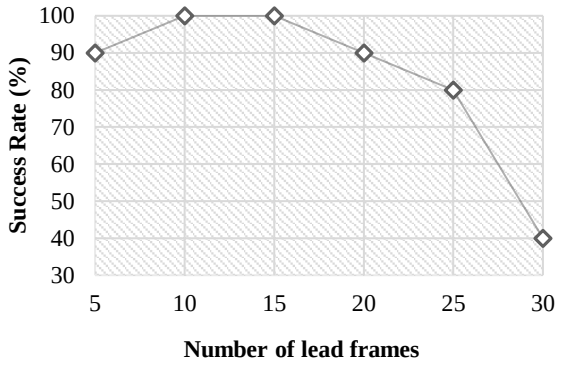


Fig. 15: Success rate of the reinforcement learning algorithm to determining the relevant frames versus the number of total lead frames.

According to the results in Fig. 15, the success rate of the reinforcement learning algorithm decreases as the number of frames increases beyond 15. Thus, it is decided to choose 25 to be the maximum number of frames inputted to the algorithm for generic applications.

4.3 Simulation case studies

The *i*TP-LfD approach is used to program the cobot for three tasks of different complexities. Cobot programming and control were implemented using MATLAB. The tasks are the following:

1. Task 1 - Drawing on an object: This is a one-object operation in which the cobot needs to draw a line on a fixed position on a book cover. The book varies position and orientation. This task is representative of other single-object tasks such as scanning, stamping, etc.
2. Task 2 - Pick-and-place: This is a two-object operation in which the cobot picks up a cube and places it in a box. The cube's position and orientation varies, whereas the box is at a fixed position.

3. Task 3 - Pick-trace-place: The cobot's path moves from one object to another, i.e. picks up a cube and places it in a box. This task is similar to Task 2 except that both objects change positions and orientations.

The inputs provided to the approach constitute of three demonstration images showing a top view of the initial outlay of the objects, and the path that the cobot follows in each demonstration to accomplish the task (Fig. 16). Given these inputs, the approach is able to learn and reproduce each of the tasks in new settings.

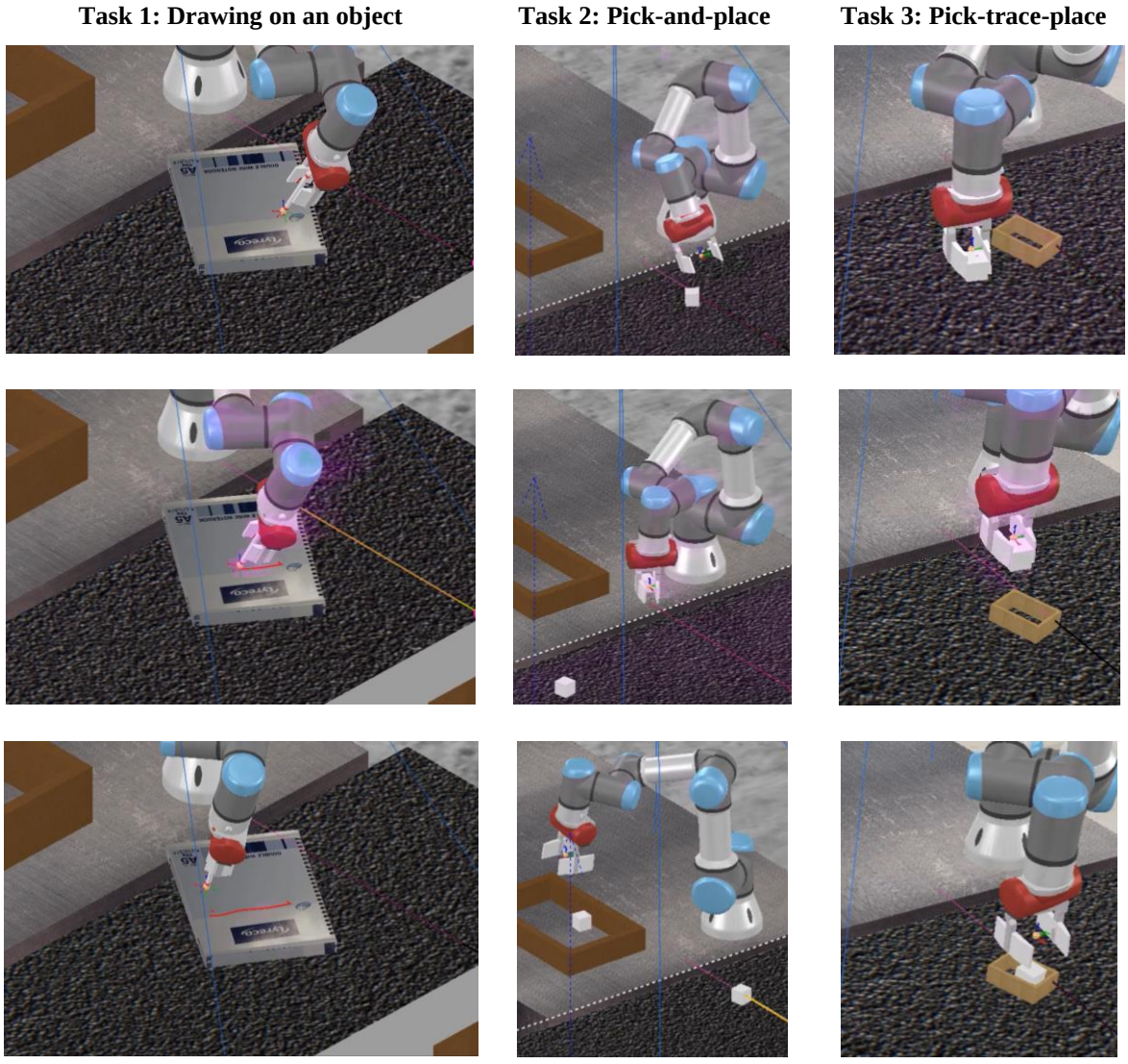


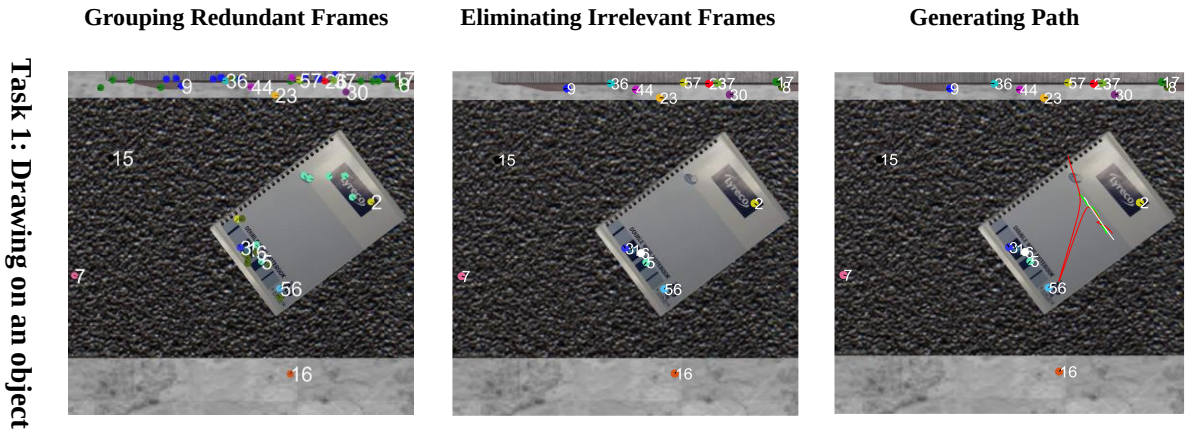
Fig. 16: Three task scenarios to validate the *i*TP-LfD approach.

Given four images of demonstrations for each scenario, three were used for training and one for validation, i.e., showing unbiased results in Fig. 17. The *i*TP-LfD approach was able to learn and reproduce all the task with only one change made to the algorithm parameters: changing the *poolsize* value. In the scenario of task 1, *poolsize*=1 since there is only one task-relevant object. However, in the scenarios of Tasks 2 and 3, *poolsize*=2, since there are two task-relevant objects (the cube and the box).

However, it is important to know that the algorithm is capable of finding the correct *poolsize* automatically (Refer to Section 4.1).

The *i*TP-LfD approach identified the relevant frames that belong to the task-relevant object(s). The path is generated using TP-GMR based on the GMMs of the relevant frames. Fig. 17 shows the results in the validation image for each task. In the first row, it shows the results of grouping redundant frames together. Dots of the same colour are redundant and are grouped together in which each group is represented by a numbered lead frame. Via experimental trials, the approach was successful in grouping redundant frames together such that the total number of lead frames is less than 25. Bringing the number of lead frames to below 25 is essential for the performance of the rest of the approach (Refer to Section 4.2). Only the lead frames are used for training the TP-GMM training.

The second row shows the relevant frames being identified after the TP-GMM training. Frames from the relevant task objects, e.g., the cube, the book, the brown box, are identified and shown in white. In the third row, the path in white is generated using TP-GMR accounting for the GMMs of relevant frame(s) only. The path in red shows the path generated using TP-GMR accounting for the GMMs of all frames. The results show that the white path is closer to the ground truth path (green) of demonstration than the red path.



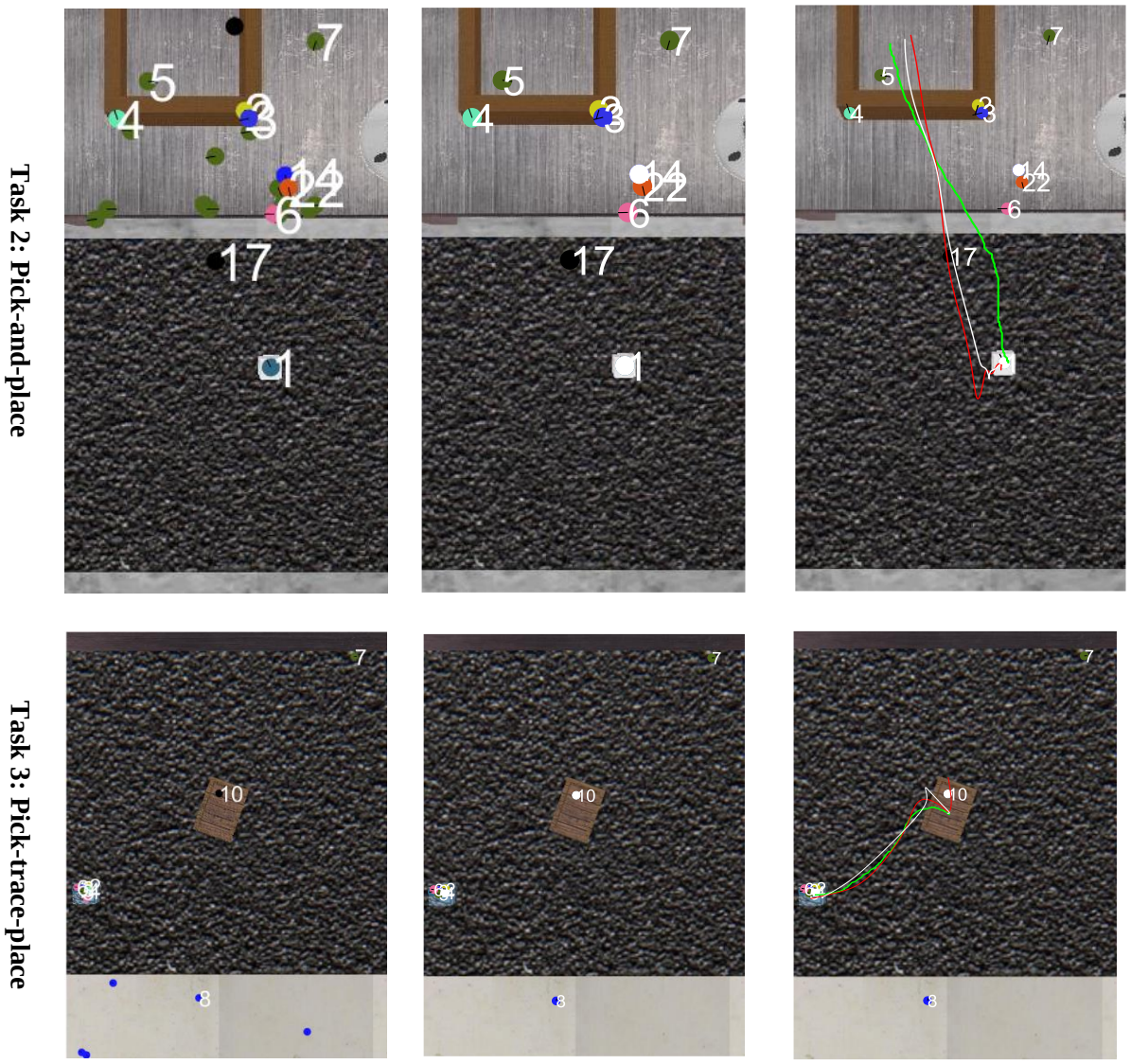


Fig. 17: Training results on the validation image. Relevant frames are autonomously detected and identified. The path reproduced using relevant frames (white) is closer to the ground truth (green) than the path reproduced using all frames (red).

Following the training, the simulation for each task reproduction was executed for 100 times. That is, 100 new images of the objects in varied positions were used to run TP-GMR, to quantitatively assess the performance of the *i*TP-LfD approach. Two different causes for reproduction errors were identified below:

1. Frame error: This includes one or more of the frames not being detected at all or being mis-located. Objects with less significant features are more prone to this error, such as a plain surface.
2. TP-GMR error: It is the error that reproduction of the TP-GMR process is not good enough. More demonstrations would help reduce this error but might limit the number of features initially detected in the algorithm, thus increasing the chance of failure. Moreover, *i*TP-LfD approach is an orientation sensitive algorithm generating the error.

Table 4 shows the occurrence percentage of each error in the 100 runs of each scenario.

Table 4: The occurrence percentage of errors in the 100 runs of each scenario.

Scenario	**Success**	**Frame error**	**TP-GMR error**
Draw on Object	100%	0	0
Pick-and-place	92%	4%	4%
Pick-trace-place	94%	5%	1%

Task 1 only has one task-relevant object, which eliminates the errors of TP-GMR. Moreover, the object, i.e., a book, has distinctive visual features such as writing and shapes. This eliminated frame errors. In Task 2, the errors of TP-GMR occurred when the cube was placed on the boundaries of the space where reproduction might fail unless more varied demonstrations were provided for training. In Tasks 2 and 3, frame errors occurred occasionally when the conveyor belt was confused for the stationary table or the wooden box respectively.

The measurements to further eliminate the above errors and improve the *i*TP-LfD approach for practical situations are discussed below.

4.4 Discussions

Detecting visual features is an essential part of the *i*TP-LfD approach. It was found that matching SURF features tends to fail in certain scenarios, especially when an object is symmetrical, when identical objects are present and/or an object is reflective. Therefore, it is recommended to use *i*TP-LfD approach when task objects are unique and non-reflective. If used for reflective objects, it is recommended to use an appropriate industrial lighting.

Finding redundant frames given real data proved challenging. The results showed that some redundant frames were detected but not all. That is, a single rigid object was divided into several objects. This is due to two main reasons:

1) Slight mis-localisations: The location of the frame is slightly inconsistent across demonstrations due to camera calibration issues, object being skewed with respect to surface, etc. We aim to account for this by adjusting a threshold for comparing distance between two frames across demonstrations. The threshold is adjusted until the total number of lead frames obtained is less than 25 (refer to Section 4.2 for justification of why the number 25 is chosen).

2) Frame outliers: When a frame is mismatched in one of the frames, it is considered an outlier. Outlier frames are not detected as redundant.

The reinforcement learning algorithm was successful in identifying relevant frames to reproduce a task. The algorithm is reliable for a certain number of frames, which is one of the reasons why redundant frames need to be grouped. It is also important to note that no tweaks in parameters were done for different scenarios, even for the one with synthetic data in Fig. 16. That is because the parameters are encoded in terms of the inputs. For example, the period (number of iterations performed before updating

the Relevancy) is chosen to be equal to the number of frames. This makes the *i*TP-LfD approach applicable in a variety of scenarios with minimal changes done by the operator.

The *i*TP-LfD algorithm currently uses 2D visual features and 2D task paths, for simplicity. In future works, we aim to leverage redundant frames to be able to learn 3D paths using 2D features. This would make it possible for a cobot to learn 3D paths such as tracing at variable heights or inclined surfaces and complex assembly tasks. Moreover, SURF features are not effective at detecting human hands since they change shapes between demonstrations. Therefore, supportive collaborative tasks [29] such as hand-over and co-manipulation cannot be learnt using *i*TP-LfD. In future works, we aim to incorporate hand detection using body part trained detection models, such as OpenPose [14], to encode the paths with respect to the human partner's hand as well.

The probabilistic nature of TP-GMM, which models the task paths, makes *i*TP-LfD suitable for tasks with path tolerance, such as applications for human-robotic collaboration and cobots. However, some algorithms in the *i*TP-LfD approach, such as the vision algorithm, are not restricted to cobots and can be integrated with other path reproduction techniques that cater for deterministic (low tolerance) paths and hence non-collaborative robots.

4.5 Implementation

In this research, physical experiments were conducted to validate the *i*TP-LfD approach using the aforementioned pick-and-place case. The hardware set-up is shown in Fig. 18.

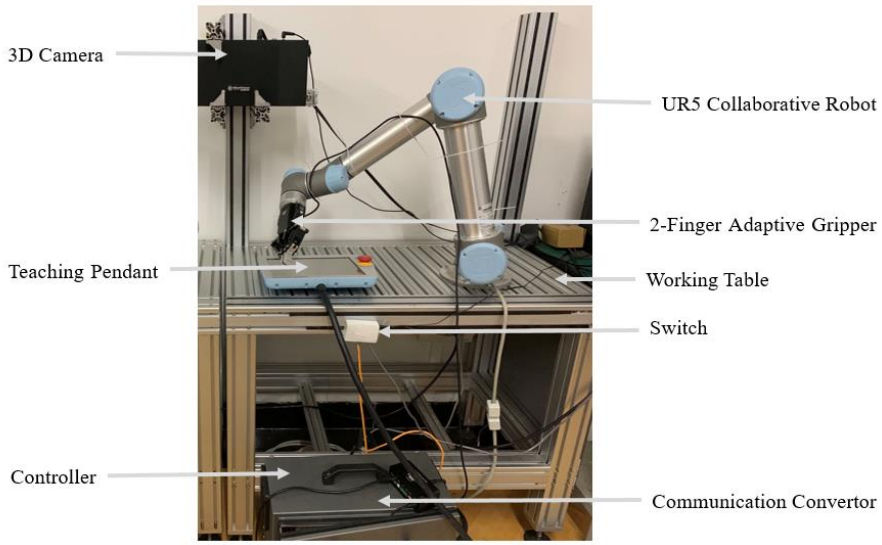
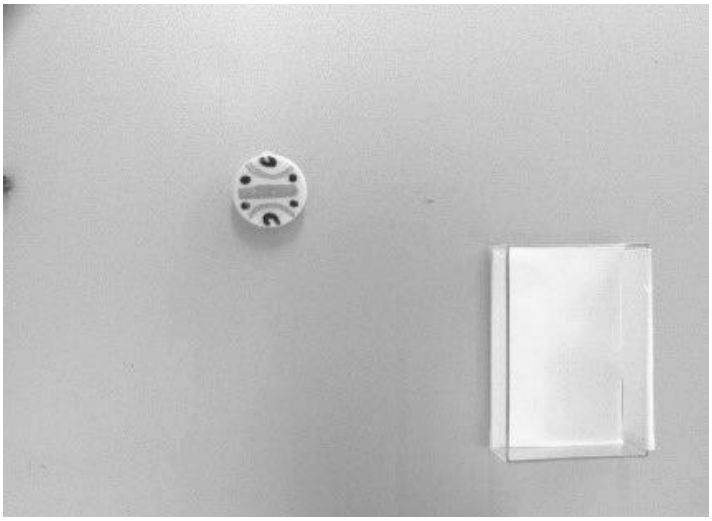
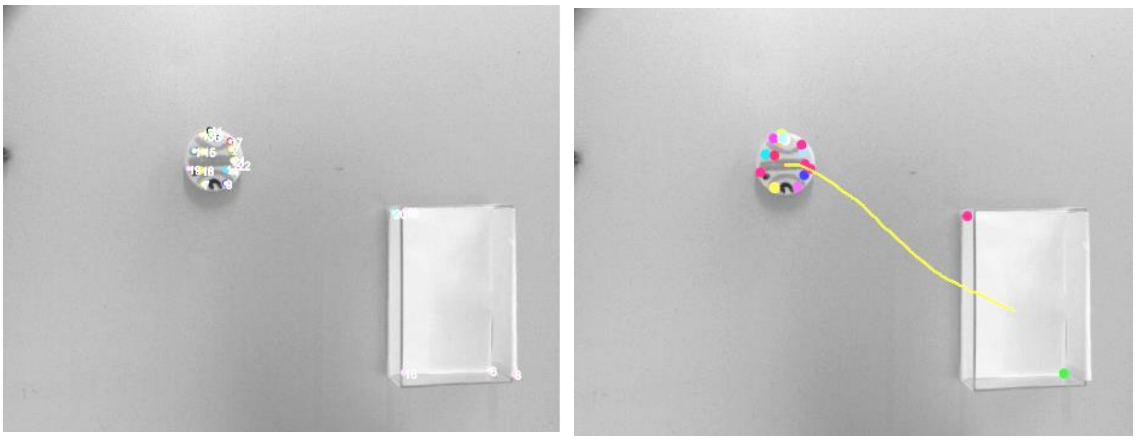


Fig. 18: The experimental setup.

The set-up includes a UR5 cobot, a 2-finger adaptive gripper, a switch, a 3D camera and a computer. Four demonstration images were captured by the 3D camera. A cylindrical object (called target-A) is picked and placed into a rectangular storage box (called des-B). The position and direction of target-A can be varied while des-B is fixed. Fig. 19 shows the image of the original setup and the demonstration paths (using one demonstration as an example).



(a) The original set-up for the demonstrations



(b) One demonstration as an example (there will be four demonstrations)

Fig. 19: Demonstrations constituting of 2D image of initial setup and the recorded demonstration path. The visual features are detected and matched between the images.

According to the *i*TP-LfD approach, the paths were used to train the TP-GMM process. Moreover, the optimal task parameters, i.e., frames, were automatically detected by the *i*TP-LfD approach. Given a new image of the setting, Fig. 20 shows the the reproduction paths generated by the TP-GMR process. Fig. 21 shows the physical implementation.

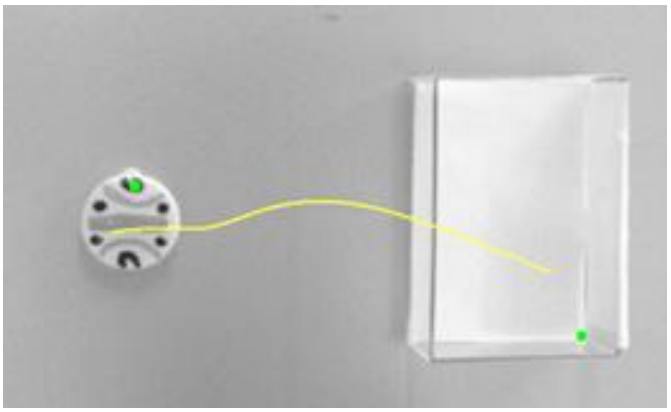


Fig. 20: Test image showing the object in a previously unseen position. The relevant frames are automatically detected and the path is reproduced successfully going from target-A to des-B.

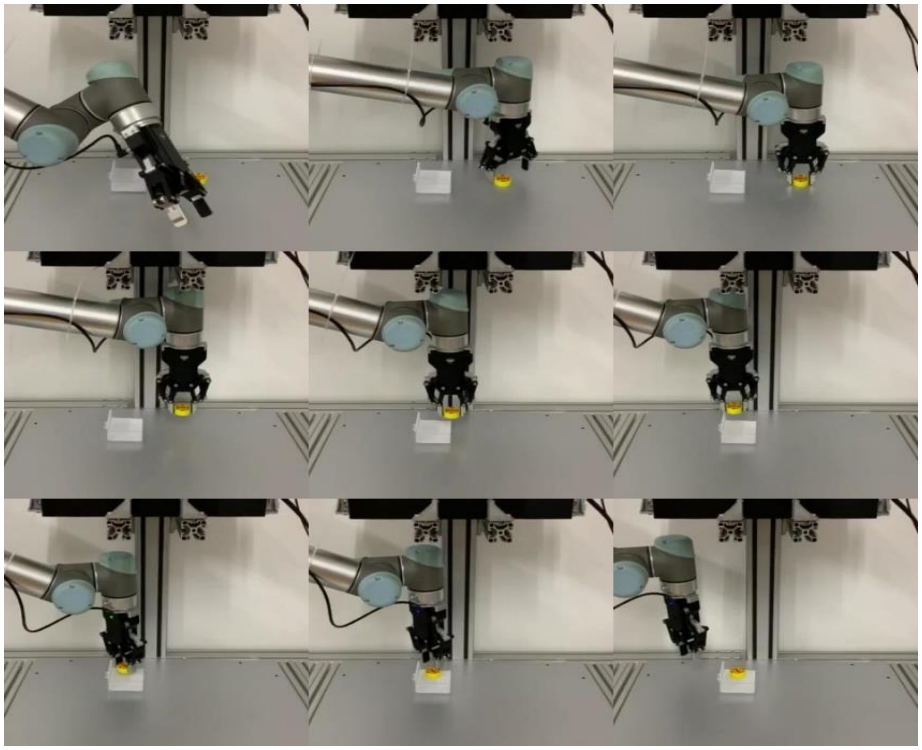


Fig. 21: The implementation of the pick-and-place process. In the setup as shown in the test image, the cobot performs the reproduction path. The cobot approaches target-A, pick it up, follows the path to the des-B and drops the object in the box.

5. Conclusions

In this paper, an *i*TP-LfD approach is presented to support cobots to adapt their movements intelligently and adaptively for a variety of dynamic industrial tasks. There are three major improvements of this *i*TP-LfD approach over developed TP-LfD approaches, including: 1) a generic visual feature extraction algorithm is designed to effectively identify frames of reference from demonstrations to regenerate the path of a cobot for dynamic settings, 2) a statistical algorithm is

developed to identify redundant frames of reference to simplify computational complexity and overcome occlusions in applications, and 3) a reinforcement learning algorithm is devised to eliminate irrelevant frames to enhance computational robustness. Case studies with different complexities show that the overall performance of the *i*TP-LfD approach are significantly enhanced in terms of adaptability and robustness.

In future works, further improvements on eliminating frame errors and TP-GMR errors will made. It aims to extend the approach to more complex industrial tasks and run a user-study for operators, eventually leading to the deployment of the approach on factory floors.

Acknowledgement

This work is funded by the PhD studentships from the Coventry University, the Unipart Powertrain Application Ltd. (U.K.), the Institute of Digital Engineering, U.K., and a research project sponsored by the National Natural Science Foundation of China (Project No. 51975444).

References

[1] A. Realyvásquez-Vargas, K. C. Arredondo-Soto, J. L. García-Alcaraz, B. Y. Márquez-Lobato and J. Cruz-García, "Introduction and configuration of a collaborative robot in an assembly task as a means to decrease occupational risks and increase efficiency in a manufacturing company," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 315-328, 2019.

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

[3] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016, p. 64.

[4] K. R. Guerin, S. D. Riedel, J. Bohren and G. D. Hager, "Adjutant: A Framework for Flexible Human-Machine Collaborative Systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, Chicago, 2014.

[5] C. Schou, R. S. Andersen, D. Chrysostomou, S. Bøgh and O. Madsen, "Skill-based instruction of collaborative robots in industrial settings," *Robotics and Computer Integrated Manufacturing*, vol. 53, pp. 72-80, 2018.

[6] F. Steinmetz, A. Wollschlager and a. R. Weitschat, "RAZER—A HRI for Visual Task-Level Programming and Intuitive Skill Parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1362-1369, 2018.

[7] C. Paxton, A. Hundt, F. Jonathan, K. Guerin and G. D. Hager, "CoSTAR: Instructing Collaborative Robots with Behavior Trees and Vision," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, 2017.

[8] F. Steinmetz and R. Weitschat, "Skill Parametrization Approaches and Skill Architecture for Human-Robot Interaction," in *Proceedings of IEEE International Conference on Automation Science and Engineering (CASE)*, 2016.

[9] T. Gašpar, M. Deniša, P. Radanovič, B. Ridge, T. R. Savarimuthu, A. Kramberger, M. Priggemeyer, J. Roßmann, F. Wörgötter, T. Ivanovska, S. Parizi, Ž. Gosar, I. Kovač and A. Ude, "Smart hardware integration with advanced robot programming technologiesfor efficient reconfiguration of robot workcells," *Robotics and Computer Integrated Manufacturing,* vol. 66, 2020.

[10] D. Vogt, S. Stepputtis, S. Grehl, B. Jung and H. B. Amor, "A system for learning continuous human-robot interactions from human-human demonstrations," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA),* Singapore, 2017.

[11] C. Perez-D'Arpino and J. A. Shah, "C-LEARN: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy," in *2017 IEEE International Conference on Robotics and Automation (ICRA),* Singapore, 2017.

[12] J. d. G. Fernandez, D. Mronga, M. Gunther, T. Knobloch, M. Wirkus, M. Schroer, M. Trampler, S. Stiene, E. Kirchner, V. Bargsten, T. Banziger, J. Teiwes, T. Kruger and F. Kirchner, "Multimodal sensor-based whole-body control for humanrobot collaboration in industrial settings," *Robotics and Autonomous Systems,* vol. 94, p. 102–119, 2017.

[13] M. Haage, G. Piperagkas, C. Papadopoulos, I. Mariolis, J. Malec, Y. Bekiroglu, M. Hedelind and D. Tzovaras, "Teaching Assembly by Demonstration using Advanced Human Robot Interaction and a Knowledge Integration Framework," *Procedia Manufacturing,* vol. 11, p. 164 – 173 , 2017.

[14] Z. Cao, T. Simon, S.-E. Wei and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Honolulu, 2017.

[15] H. Jin, Q. Chen, Z. Chen, Y. Hu and J. Zhang, "Multi-LeapMotion sensor based demonstration for robotic refine table-top-object manipulation task," *CAAI Transactions on Intelligence Technology ,* vol. 1, pp. 104-113, 2016.

[16] A. M. Ghalamzan and M. Ragaglia, " Robot learning from demonstrations: Emulation learning in environments with moving obstacles," *Robotics and Autonomous Systems,* vol. 101, p. 45–56, 2018.

[17] D. A. Duque, F. A. Prieto and J. G. Hoyos, "Trajectory generation for robotic assembly operations using learning by demonstration," *Robotics and Computer Integrated Manufacturing,* vol. 57, pp. 292 - 302 , 2019.

[18] K.-B. Park, M. Kim, S. H. Choi and J. Y. Lee, "Deep learning-based smart task assistance in wearable augmented reality," *Robotics and Computer Integrated Manufacturing ,* vol. 63, 2020.

[19] Y. Gu, W. Sheng, C. Crick and Y. Ou, "Automated assembly skill acquisition and implementation through human demonstration," *Robotics and Autonomous Systems,* vol. 99, pp. 1-16, 2018.

[20] K. Wang, D. Liu, Z. Liu, G. Duan, L. Hu and J. Tan, "A fast object registration method for augmented reality assembly withsimultaneous determination of multiple 2D-3D correspondences," *Robotics and Computer Integrated Manufacturing,* vol. 63, 2020.

[21] T. Alizadeh and N. Karimi, “Exploiting the task space redundancy in robot programming by demonstration,” in *Proceedings of 2018 IEEE International Conference on Mechatronics and Automation* , Changchun, 2018.

[22] T. Alizadeh and M. Malekzadeh, “Identifying the relevant frames of reference in programming by demonstration using task-parameterized Gaussian mixture regression,” in *Proceedings of the 2016 IEEE/SICE International Symposium on System Integration*, Sapporo, 2016.

[23] A. Sena, B. Michael and M. Howard, “Improving Task-Parameterised Movement Learning Generalisation with Frame-Weighted Trajectory Generation,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019)*, Macau, 2019.

[24] Y. Huang, J. Silverio, L. Rozo and D. G. Caldwell, “Generalized Task-Parameterized Skill Learning,” in *Proceedings of 2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, 2018.

[25] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv*, 2018.

[26] S. A. K. Tareen and Z. Saleem, “A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK,” in *Proceedings of Conference: 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET 2018)*, Sukkar, 2018.

[27] H. Bay, T. Tuytelaars and L. V. Gool, “SURF: Speeded Up Robust Features,” in *Proceedings of European Conference on Computer Vision*, 2006.

[28] M. Muja and D. G. Lowe, “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration,” in *Proceedings of International Conference on Computer Vision Theory and Applications*, Lisboa, 2009.

[29] S. E. Zaatari, M. Marei, W. Li and Z. Usman, “Cobot programming for collaborative industrial tasks: An overview,” *Robotics and Autonomous Systems*, pp. 162-180, 2019.