Ring Gaussian Mixture Modelling and Regression for Collaborative Robots

El Zaatari, S., Li, W. & Usman, Z.

Original citation & hyperlink:
El Zaatari, S, Li, W & Usman, Z 2021, 'Ring Gaussian Mixture Modelling and Regression for Collaborative Robots', Robotics and Autonomous Systems, vol. 145, no. November 2021, 103864.
https://dx.doi.org/10.1016/j.robot.2021.103864

DOI 10.1016/j.robot.2021.103864
ISSN 0921-8890

Publisher: Elsevier

Ring Gaussian Mixture Modelling and Regression for Collaborative Robots

Shirine El Zaatari[a], Weidong Li[a,b*], Zahid Usman[c]

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

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

[c] Rolls-Royce plc, UK

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

Abstract

Task Parametrised Gaussian Mixture Modelling and Regression (TP-GMM/R) is an eminent algorithm to enable collaborative robots (cobots) to adapt to new environments intuitively by learning robotic paths demonstrated by humans. Task parameters in the TP-GMM/R algorithm, i.e., frames associated with demonstration paths, are considered to have orientations by default. This requirement, however, limits the range of applications that TP-GMM/R can support. To address the issue, in this paper, a novel ring Gaussian (*r*Gaussian) is defined to cater for orientation-less frames, and an improved TP-GMM/R algorithm based on *r*Gaussians is developed to improve the adaptability and robustness of the algorithm. In the improved algorithm, firstly, kernels are incorporated to enable Gaussians encoding points from all demonstrations, and criteria are devised to judge a frame to be oriented or orientation-less. Then, improved Gaussian mixture regression that caters for *r*Gaussians and orientation-less frames is developed to generate regression paths adaptable to complex environments. Finally, a series of case studies are used to benchmark the improved TP-GMM/R algorithm with the conventional TP-GMM/R algorithm under different conditions. Quantitative analyses are conducted in terms of smoothness, efficiency and reachability. Results show that the improved algorithm outperformed the conventional algorithm on all the cases.

Keywords: Learning from Demonstration, Gaussian Mixture Model, Gaussian Mixture Regression, Collaborative robots (Cobots)

Annotations in this paper:

Dimensions:	
N (n used as index)	Number of demonstrations used to train the TP-GMM/R algorithm
J (j used as index)	Number of task parameters, i.e., frames of reference
K (k used as index)	Number of Gaussians
T (t used as index)	Number of time steps, i.e., number of points along a demonstration path
Elements:	
$b_{(j)}^{(n)}=[bx,by]_{(j)}^{(n)}$	x-y position coordinates of Frame j in Demonstration n
$\alpha_{(j)}^{(n)}$	Rotation angle of Frame j in Demonstration n
$A_{(j)}^{(n)}$	2x2 rotation matrix for Frame j in Demonstration n by an angle $\alpha_{(j)}^{(n)}$
$p_{(t)}^{(n)}=[t,x,y]_{(t)}^{(n)}$	Point at time step t along path of Demonstration n
$p_{(t,j)}^{(n)}=[t,x,y]_{(t,j)}^{(n)}$	Point at time step t along path of Demonstration n relative to Frame j

Gaussian Distribution:	
$\mathcal{N}(\mu_{(k,j)}, \Sigma_{(k,j)}, w_{(k,j)})$	The k^{th} Gaussian of path points with respect to Frame j
$\mu_{(k,j)}=[\mu t, \mu x, \mu y]_{(k,j)}$	Centre of the k^{th} Gaussian of Frame j; it is the mean value of all three dimensions of path points (t, x, y)
$\mathcal{N}(\mu_{GMR(t,j)}, \Sigma_{GMR(t,j)})$	The t^{th} Gaussian of path points with respect to Frame j
$\mu GMR(t,j)=[\mu_x GMR, \mu_y GMR]_{(t,j)}$	Centre of the t^{th} Gaussian of Frame j; it is the mean value of two dimensions of path points (x, y)
Ring Gaussian (*r*Gaussian) Distribution:	
$\mathcal{N}_r(\mu_{r(k,j)}, \Sigma_{r(k,j)})$	The k^{th} ring Gaussian of path points with respect to Frame j
$\mu_{r(k,j)}=[\mu t, \mu r]_{(k,j)}$	Centre of the k^{th} ring Gaussian of Frame j; it is the mean value of two dimensions of path points (t, r)
$\Sigma_{r(k,j)}$	2x2 covariance of the k^{th} Gaussian of Frame j
$\mathcal{N}_r(\mu_r GMR(t,j), \sigma_r GMR(t,j))$	The t^{th} ring Gaussian of path points with respect to Frame j
$\mu_r GMR(t,j)$	Centre of the t^{th} ring Gaussian of Frame j; it is the mean value of r
$\sigma_r GMR(t,j)$	Variance of r in the t^{th} ring Gaussian of Frame j

1. Introduction

1.1 Introduction to Learning from Demonstration

In modern factories, interest has been growing in adopting collaborative robots (cobots) as a step towards boosting ergonomics and enabling mass customisation in manufacturing [1]. Cobots are able to work alongside humans owing to their built-in safety features such as power and speed limiting and the absence of trap points [2]. However, to reach their full potential in being intelligent "collaborators" with human operators, cobots need to be equipped with two functions: flexible behaviour and intuitive programming. The flexible behaviour would enable a cobot to adjust its motions and task plans based on the human's as well as adapting to the unpredictable positions of parts and tools. This allows the cobot to achieve full collaboration [3], i.e., performing a task together with a human collectively, as opposed to merely working in close proximities with him/her. A cobot should also be intuitively programmed so that operators can easily create or adjust the cobot's program on the fly. Having operators involved in the programming of a cobot provides the operators with intuition about the cobot's behaviour which facilitates collaboration. Moreover, it allows the operators to alter the program depending on their needs and interact with the cobot dynamically during the task if necessary.

Learning from Demonstration (LfD) is one of the prominent algorithms that combine the above two functions [4]. It has been applied in several industrial robotic operations including assembly [5], grasping [6] [7], pick-and-place [8] and polishing [9]. Moreover, LfD has been used to program cobots for human-robotic collaboration tasks, such as collaborative assembly [10] and collaborative handing [11]. LfD means that operators intuitively teach a cobot a task by providing demonstrations of the task

being done. These demonstrations are often in the form of paths (or trajectories) that should be traced by a cobot to perform a task. The cobot does not simply record and repeat the task, but rather learns and generalises the task to perform it even after changes in the initial environment of the task. Demonstration paths can be modelled in a variety of formats, the most popular being Gaussian Mixture Model (GMM). According to the central limit theorem, normal distributions, i.e., GMM, can effectively model complex systems with the least amount of prior knowledge needed [12]. Hidden Markov Model (HMM), which is another popular probabilistic learning from the demonstration technique, models spatial and temporal variability. However, since HMM models the sequence between a finite discrete number of states, it is intrinsically suited for task-level learning rather than motion-level learning [13]. When HMM is used to model continuous paths, the paths are first modelled using a GMM and each Gaussian component is a GMM state. Then HMM models the probability of switching between Gaussian components that intrinsically accounts for temporal variability. For modelling robotic paths, the accuracies of GMM and HMM are comparable but the latter is more complex in the model [14]. Also, the path modelled using GMM is smoother than that using HMM, which ensures the better kinetic features for robots [14]. Thus, GMM is more adaptive to support the requirements of various industrial applications such as varying impedance [15], arrayed motions [16] and geometrical joint constraints [17]. Based on the above observations, GMM is chosen as the modelling representation in this research.

A path is then regenerated from GMM using a complimentary process known as Gaussian Mixture Regression (GMR). GMR relies on the properties of normal distributions (linear transformations and conditioning) to rapidly generate a continuous, infinitely differentiable path [13], by deriving a regression function based on the conditional probability between demonstration paths. For example, a demonstration could include a series of path points, characterised by a time step t and a position in space. The regression can be performed such that time steps are user-controlled, and point positions are regressed accordingly. That allows the production of dynamic results based on environmental conditions.

1.2 Introduction to TP-GMM/R

Task Parametrised GMM (TP-GMM) is an extension of GMM [13]. In GMM, path point positions are defined and modelled with respect to a global frame of reference. However, in TP-GMM, they are modelled as multiple GMMs with respect to multiple task parameters, i.e., frames of reference (to simplify, frames of reference are called frames in the rest of this paper). A frame represents the position and orientation of an object/location in the environment. TP-GMM allows a cobot to learn complex path dependencies to different objects/locations, such as prioritising the effect of one frame on the path over another frame's effect on it. Then, in a new environment, given new positions and orientations of frames, Task Parametrised Gaussian Mixture Regression (TP-GMR), which is similar to GMR in purpose, automatically reproduces a new path for a new scenario, prioritising the effects of different frames for different demonstration paths.

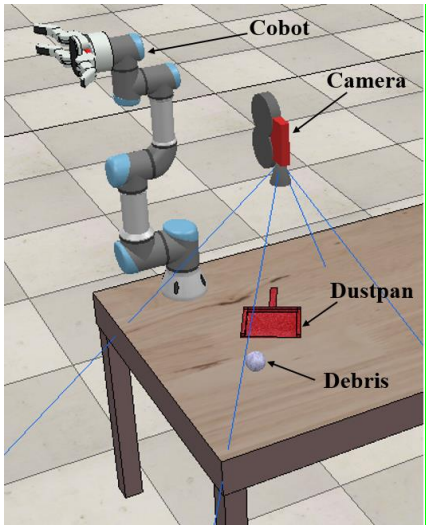


Fig. 1: A scenario in which a cobot needs to brush a piece of debris onto a dustpan.

For example, consider a task in which a cobot has to brush a piece of debris onto a dustpan (shown in Fig. 1). An operator records several demonstrations, in which he/she traces paths that the cobot needs to perform to successfully move the debris onto the dustpan (Fig. 2). Each path is formed of a series of *T* points and each point consists of a time step *t* and its position. Assume Frame 1 is associated with the debris and Frame 2 with the dustpan. In each demonstration, the positions and orientations of both frames change purposefully to provide variability, which is necessary for TP-GMM.

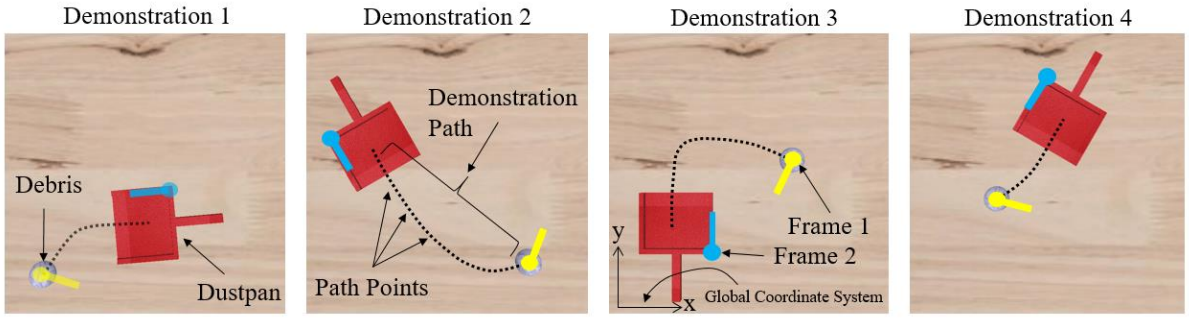
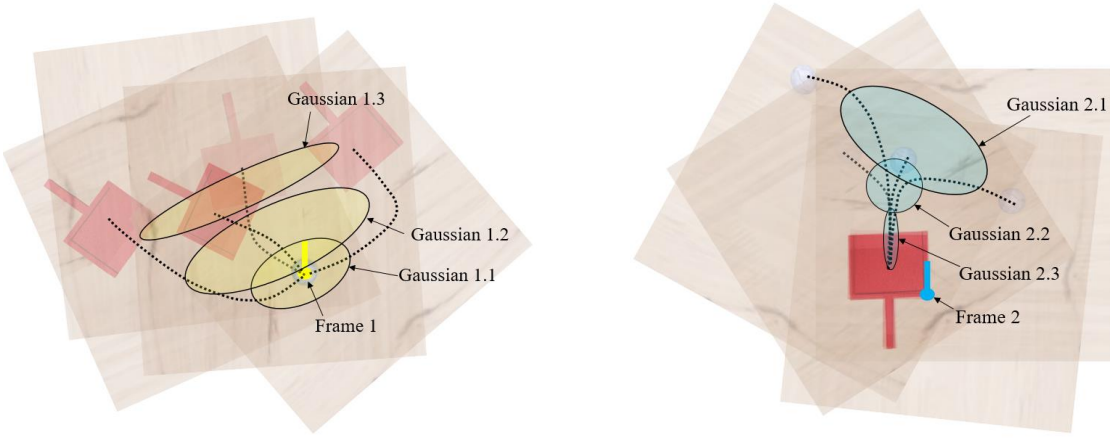


Fig. 2: Four demonstration recordings of the cobot's paths for brushing the debris onto the dustpan.

The demonstration paths are transformed from the global coordinate system to the local coordinate system of each frame. For each frame, the path points from the demonstrations are modelled as a GMM (for more mathematical details, please refer to Section 3). Each GMM is a mixture of Gaussian distributions (Gaussian for simplicity in the rest of the paper), each characterised with a mean and a covariance matrix. (e.g., In Fig. 3, Gaussians 1.1, 1.2 and 1.3 are modelled as a GMM for Frame 1; Gaussians 2.1, 2.2 and 2.3 are modelled as a GMM for Frame 2). This process is known as TP-GMM.

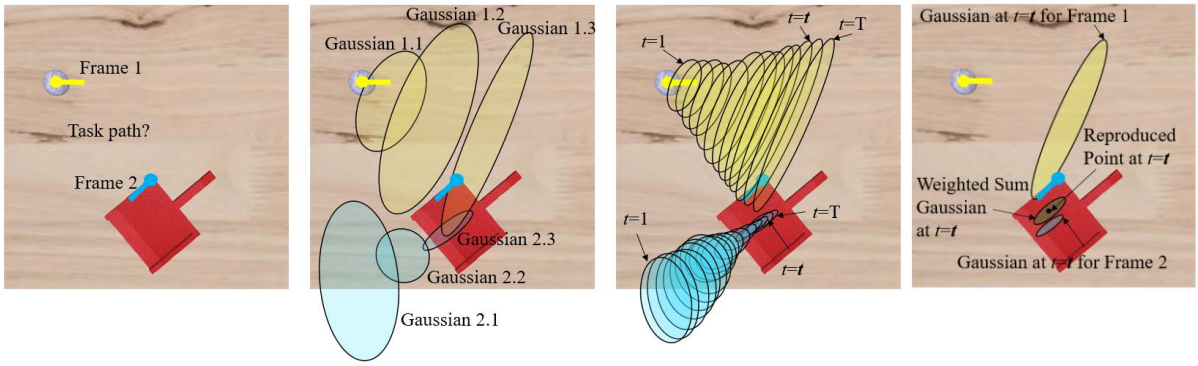


(a) The GMM composed of three Gaussians modelling path points when they are aligned with respect to Frame 1 (associated with the debris).

(b) A GMM composed of three Gaussians modelling path points when they are aligned with respect to Frame 2 (associated with the dustpan)

Fig. 3: Examples for frames, Gaussians and GMM.

TP-GMR is illustrated using the examples in Fig. 4. Based on demonstrations, as the positions and orientations of the dustpan and the debris change, the cobot should reproduce a new path for this new scenario accordingly (Fig. 4(a)). Obtained GMMs for each frame are lined up depending on the frames' new positions and orientations in this scenario (Fig. 4(b)). For each frame, GMR is performed, where a path point distribution, i.e. Gaussian, for each time step t is sampled from the GMM (Gaussians under different time steps are shown in Fig. 4(c)). In order to reproduce the new path, the Gaussians of both frames need to be considered. For a time step t, a product of Gaussians is performed between the Gaussians of both frames. The product results in a new Gaussian, whose mean is the reproduced path point at time t (Fig. 4(d)). The weight for each Gaussian in the sum is equivalent to the inverse of its covariance matrix. Thus, distributions that have higher covariance possess less priority in the sum. The resultant Gaussian is more likely to tend towards Gaussians with smaller covariance, which means a frame with respect to which the path varies less is prioritised. The above process is known as TP-GMR.



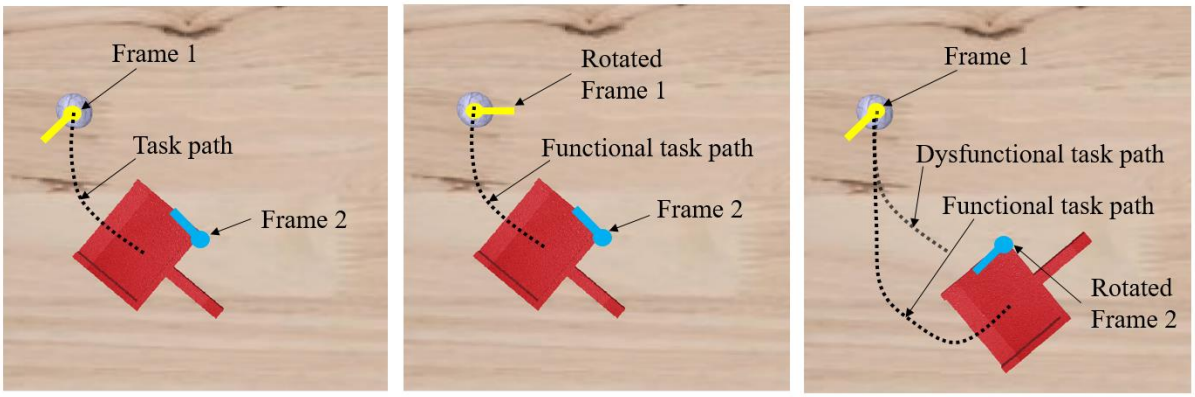
(a) To generate a new path (b) Gaussians for frames (c) Gaussians along time step t (d) Reproduced path points

Fig. 4: The main steps of the TP-GMM/R algorithm.

1.3 Limitations of TP-GMM/R

To understand the main limitation of TP-GMM/R to be tackled in this paper, firstly, two types of frames, i.e., "oriented frames" (OFs) and "orientation-less frames" (OLFs), are explained. OFs are

frames whose orientations are relevant to a demonstration. For example, in Fig. 5(a), the frames and the demonstration path are created by a human operator. In Fig. 5(b), if the orientation of the debris changes, the demonstration path does not have to change to perform the task goal. That is, because of the symmetrical shape of the debris and the nature of the task, the cobot can still successfully move the debris from any of the debris' sides. Thus, Frame 1 is considered as an OLF, as its orientation is irrelevant to the demonstration path. In Fig. 5(c), since the debris can only enter the dustpan from one direction, if the orientation of the dustpan changes, the demonstration path has to change as well. Therefore, the frame associated with the dustpan, Frame 2, is called an OF. That is, when the orientation of Frame 2 changes, the demonstration path needs to be adjusted to perform the task goal properly.



(a) (b) (c)

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

Fig. 5: Examples illustrating OLFs and OFs.

As discussed in the previous subsection, the TP-GMM/R algorithm models demonstrations with respect to each of the frames, taking into account the frame's positions and orientations. Therefore, the modelling process inherently encodes a relation between the demonstration paths and the orientation of a frame. However, in a situation that a demonstration path is independent of the orientation of a frame (e.g., OLF), the model resulting from TP-GMM/R does not describe the path-frame relation accurately, but rather becomes falsely biased towards what is presented in the demonstrations. For example, in Fig. 4(b), Gaussian 1.2 and 1.3 imply that the path tends to a particular side of Frame 1 while the orientation-less nature of the frame and flexibility are lost. That is due to the fact that the demonstrations slightly tend towards one side of the frame. However, Frame 1 is orientation-less, so that the path points are uniformly likely to occur around the frame. Therefore, to improve the performance and flexibility of TP-GMM/R in processing OLFs, a new model is required to reflect the uniform probability of path points of demonstrations around OLFs, regardless of the frame's orientation. This is particularly important in some applications such as pick-and-place tasks, where it is likely for the picked object or the drop location to be an OLF.

Another limitation of the TP-GMM algorithm is that, when modelling the distribution of demonstration path points, the algorithm does not account for which demonstration each point belongs to. Instead, all path points are regarded equally by the algorithm regardless of whether they belong to the same or different demonstration paths. Therefore, when the GMM is fitted to the points, there is a risk that a Gaussian in the GMM models points from a particular demonstration (if they happen to be clustered together) rather than points from all demonstrations. However, the goal of TP-GMM is to learn tasks based on the variations of demonstrations that an operator provides [13]. Therefore, to model the distributions of path points based on the variations in the demonstrations provided, the TP-GMM algorithm should be aware of which demonstration each path point belongs to. Otherwise, there might be a case of disproportional demonstration modelling or overfitting. The situations are depicted further in Section 3.

To overcome the above limitations, this paper presents an improved TP-GMM/R algorithm (for the purpose of differentiation, in the following, the TP-GMM/R algorithm introduced earlier are called "conventional"). The algorithms include the following characteristics and innovative contributions:

1) In this research, frames are defined as OFs and OLFs to facilitate LfD-based cobots in supporting complex applications effectively. Based on a set of newly developed Gaussian criteria, an algorithm is designed to intelligently identify OFs and OLFs;
2) On the basis of OLFs, an innovative Gaussian model called ring Gaussian (*r*Gaussian) is developed to overlook the orientations of frames and provide superior results over the conventional TP-GMM algorithms when encoding demonstrations with respect to OLFs. The conventional TP-GMM/R algorithm, which was designed for processing OFs, is enhanced here as the improved algorithm by incorporating the *r*Gaussian;
3) A new cost function in the TP-GMM/R algorithm is devised to ensure that the resultant GMMs are able to model all demonstration paths equally, thereby preventing GMMs from overfitting some demonstration paths more than others;
4) Case studies were used to validate the improved algorithms presented in this paper quantitatively to prove their robustness and adaptability.

The rest of the paper is organised as follows. Section 2 summarises related works and identifies research gaps. Section 3 introduces the fundamentals of the conventional TP-GMM/R algorithm in more detail. On this basis, Section 4 explains the methodology of the improved TP-GMM/R algorithm proposed in this paper. Section 5 presents how the algorithms are evaluated on synthetic data and real-life industrial scenarios. Section 6 concludes the research and outlines the future works.

2. Related Works

TP-GMM/R has become an increasingly popular research topic of LfD in supporting various applications, from collaborative assembly [15], collaborative object movement [16], daily kitchen tasks [17], etc. In this section, the related research will be briefly reviewed from the aspects of demonstration

data acquirement, task parameter identification, and performance improvement measurements on TP-GMM/R.

For LfD, a demonstration usually refers to a path for a cobot to manipulate an object from a start point to an end point for moving, assembling, etc. There are multiple ways for a human operator to record a demonstration to be learnt by a cobot. Kinaesthetic teaching [18] [19], observational teaching [10] and teleoperation [20] are the most common methods of recording demonstrations. Fischer et al. provided a comparative study on the methods of acquiring demonstrations [21], highlighting the pros and cons of each method. Some works combined multiple methods to improve the performance of LfD. For example, Yang et al. created a bracelet worn by an operator and attached to the end effector of a cobot [22]. This enables the operator to wear the bracelet and intuitively move the cobot to perform a task as if he/she is performing it himself/herself. Ogenyi et al. created an algorithm through which a cobot watches and learns from an operator performing a task [23]. Then, the operator corrects and improves the cobot's learnt demonstration using kinaesthetic teaching.

Task parameters usually refer to the positions and orientations of frames in demonstrations. Some research works localised frames using image processing methods, such as colour/shape segmentation [15] [24] or contour identification [25]. However, these methods require well-controlled environments and objects. More sophisticated computer vision algorithms such as Conventional Neural Networks (CNNs) were used to detect task parameters from a given dataset of objects [11] [26]. However, this might be inconvenient for fast-changing industrial applications, especially in mass customisation, as retraining the algorithm continuously for new objects is time-consuming. Moreover, some works used invasive markers to make detection easier and more reliable, such as motion capture bulbs in [10] or sticker markers in [27] [28]. However, in industrial scenarios, invasive markers are undesirable on commercial parts or tools. In our previous research, an algorithm was devised to detect generic visual features and choose the most suitable ones amongst them to identify task parameters [20].

Meanwhile, various research works were conducted to improve the performance of the conventional TP-GMM/R [13]. Some researchers undertook demonstration-based improvements. For example, Hu and Kuchenbecker designed TP-GMM/R to program collaborative object movement but iteratively suggested adding demonstrations to improve performance [16]. Cao et al. used GMM/R iteratively to program robotic motions [29]. After each GMR, if a collision occurs, a human operator corrects the path and retrains GMM. Willibald developed an interactive learning system in which a robot automatically detects new tasks that require new demonstrations [30]. However, these demonstration-based methods are not a suitable solution for improving the performance of TP-GMM/R in the case of OLFs. Such methods will still require a human operator to record demonstrations in which paths vary in all directions around an OLF. Even though a set of demonstrations will fully describe the possible paths, the resultant GMM that models these demonstrations could still fail. That is, the GMM will either be biased towards one direction of the frame, or will be centred on the frame, both of which are not accurate representations of the paths.

Other researchers worked on frame-based solutions to improve the performance of TP-GMM/R. For example, Sena et al. calculated the importance scores of frames to amplify the weights of relevant frames during a portion of a demonstration [31]. Based on that, the TP-GMM/R algorithm can provide meaningful results even when the newly observed positions of frames are far from the previous demonstrations. Vidaković et al. designed an algorithm that classified task parameters/frames [32]. Some frames are classified as attractors and some as obstacles. A cost function was designed to ensure that obstacles are avoided and attractors are approached. Moreover, the cost function can also maintain a specific relative orientation of path points with respect to other frames. Silverio et al. designed new task parameters that help identify positional and/or orientation constraints as well as configurational constraints [33]. Instead of considering task parameters to be positions of objects in space, task parameters are Jacobians describing the absolute pose of the bimanual humanoid's end effector as well as their pose relative to the humanoid. These different task parameters are projected onto the configurational joint space of the humanoid. This allows the robot to learn complicated tasks that automatically toggle between positional/orientation and absolute/relative constraints of its end effector. Their work, however, neglects information about the positions of other objects in the same space so it does not tackle the problem of OLFs from its root cause.

To tackle the TP-GMM/R's problems related to OLFs, an OLF requires a different Gaussian model than the conventional one to accurately describe the nature of demonstration paths. None of the previous works to the best of our knowledge, tackled this problem from the TP-GMM/R-based point of view. Therefore, in this paper, the following research questions will be answered:

- How can TP-GMM/R effectively model the angular variability of demonstrations with respect to OLFs?
- How can TP-GMM/R successfully cater for scenarios having a mixture of OLFs and OFs?
- How can OLFs be automatically identified and catered for with minimal user interference?

3. Fundamentals of the TP-GMM/R Algorithm

This paper presents an improved TP-GMM/R algorithm for a cobot to carry out tasks under complex applications where there are not only oriented frames (OFs) but also orientation-less frames (OLFs). The major concepts and mathematical backgrounds of the conventional TP-GMM/R algorithm is introduced first, which will pave a base for the improved algorithm to be presented in Section 4.

3.1 Demonstrations, Gaussians and GMM

To implement TP-GMM/R, firstly, N demonstration paths are recorded. Demonstration n ($n \in \mathbb{N}\{1, \ldots, N\}$) is made up of a path $P^{(n)}$ and J frames. $P^{(n)}$ consists of T path points, and each point $p_{(t)}{}^{(n)}$ in the path is defined by time step t ($t \in \mathbb{N}\ \{1, \ldots, T\}$) and a position. Time step t indicates the time order of a point along a demonstration path. The coordinates x-y indicate the position of a path point, Frame j ($j \in \mathbb{N}\ \{1, \ldots, J\}$) is defined by a 2D position vector $b_{(j)}{}^{(n)}=[bx,by]_{(j)}{}^{(n)}$ and a 2x2 rotation matrix $A_{(j)}{}^{(n)}$ for

angle $\alpha_{(j)}^{(n)}$ with respect to the global coordinate system (the global coordinate system is defined arbitrarily as it would not affect the calculations). In Demonstration *n*, the frames may vary in position and orientation. Fig. 6 is an example to visualise the above concepts.

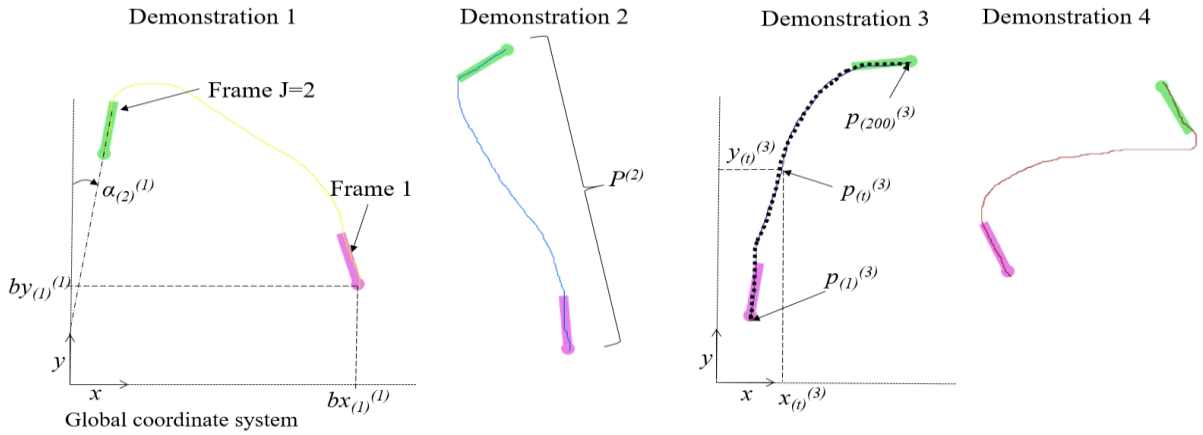
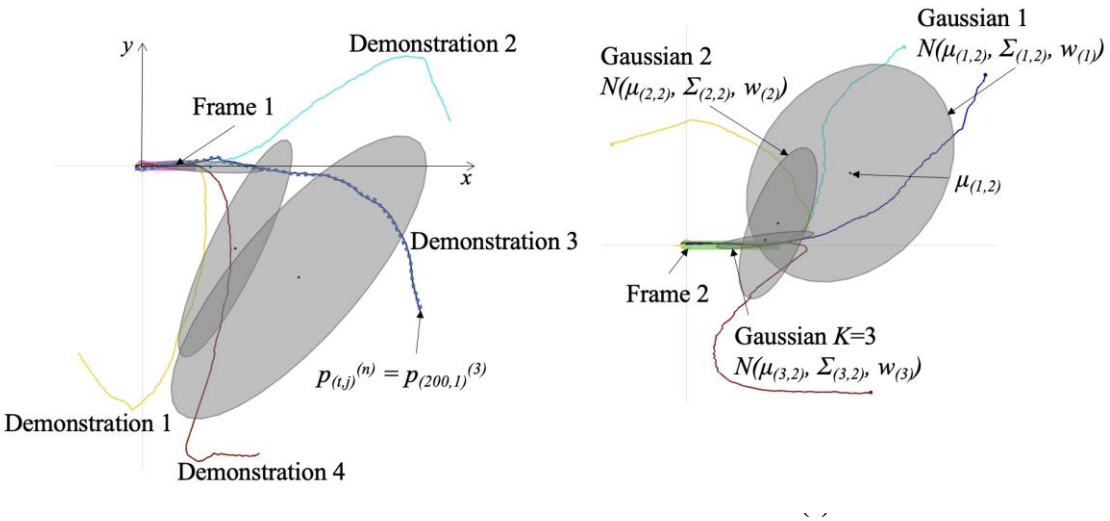


Fig. 6: Examples of paths going from Frame 1 to Frame 2. The diagram details the modelling variables, where *N*=4, *T*=200 and *J*=2.

Once the paths and task parameters are detected, data pre-processing is performed before TP-GMM is executed. That is, the x-y coordinates of each point $p_{(t)}^{(n)}$ in a demonstration are transformed from the global coordinate system to the local coordinate system of Frame *j*, forming a new point $p_{(t,j)}^{(n)}$ using the following equation:

$$p_{(t,j)}^{(n)} = [x\ y]_{(t,j)}^{(n)} = \left(A_{(j)}^{(n)}\right)^{-1}\left([x\ y]_{(t)}^{(n)} - [bx\ by]_{(j)}^{(n)}\right) \quad (1)$$

where $[x \quad y]_{(t)}^{(n)}$ are the coordinates of $p_{(t)}^{(n)}$; $[bx \quad by]_{(j)}^{(n)}$ is the position coordinates of Frame *j* in Demonstration *n*; $[x \quad y]_{(t,j)}^{(n)}$ are the coordinates of $p_{(t,j)}^{(n)}$; $A_{(j)}^{(n)}$ is the rotation matrix for Frame *j* in Demonstration *n* by Angle $\alpha_{(j)}^{(n)}$. This results in a new data set used for training TP-GMM constituting of $N \times T$ path points. The resultant paths are shown in Fig. 7.



(a) Paths are aligned with respect to Frame 1; (b) Paths are aligned with respect to Frame 2. For each frame, a GMM with K Gaussians (K=3 here) fitted to model the distribution of path points with respect to each frame.

Fig. 7: Examples of demonstrations and the GMM obtained for each frame constituting of K Gaussians each.

A GMM is a mixture of K Gaussians optimised to describe the distribution of path points in demonstrations for each frame. Gaussians are calculated simultaneously for all frames in the process as TP-GMM. Fig. 7 shows three Gaussians from each frame respectively modelling certain portions of demonstration paths.

In the coordinate system of Frame j, Gaussian k ($k \in \mathbb{N}\ \{1, \ldots, K\}$) is defined as $\mathcal{N}(\mu_{(k,j)}, \Sigma_{(k,j)}, w_{(k)})$, where $\mu_{(k,j)}$ is the mean and $\Sigma_{(k,j)}$ is the covariance matrix of Gaussian k, and $w_{(k)}$ is the weight of Gaussian k contributing to a GMM with respect to other Gaussians.

The covariance matrix $\Sigma_{(k,j)}$ is a 3x3 matrix describing the covariance between the two coordinates and time step t of each path point. For example, C_{tt} is the variance of the dimension t. C_{tx} is the covariance of dimension t with respect to dimension x. It is equal to C_{xt}, which is the covariance of dimension x with respect to t. Other symbols in the following equation are defined in the similar means.

$$\Sigma_{(k,j)} = \begin{bmatrix} C_{tt} & C_{xt} & C_{yt} \\ C_{tx} & C_{xx} & C_{yx} \\ C_{ty} & C_{xy} & C_{yy} \end{bmatrix} \tag{2}$$

The direction and size of a Gaussian are described by the Eigen vectors [v_1, v_2] and Eigen values [e_1, e_2] of Σ_{xy} (shown in Equation (3)), respectively. The Eigen vectors [v_1, v_2] and Eigen values [e_1, e_2] are calculated in Equations (4) and (5).

$$\Sigma_{xy} = \begin{bmatrix} C_{xx} & C_{yx} \\ C_{xy} & C_{yy} \end{bmatrix} \tag{3}$$

$$\Sigma_{xy}\, v_1 = e_1 v_1 \tag{4}$$

$$\Sigma_{xy}\, v_2 = e_2 v_2 \tag{5}$$

In Fig. 8, the Gaussian's two axes have basis vectors [v_1, v_2], and v_1 is the basis vector in the direction of the Gaussian's long axis. These vectors are scaled by the scalar values [e_1, e_2] to form the size and shape of the Gaussian.

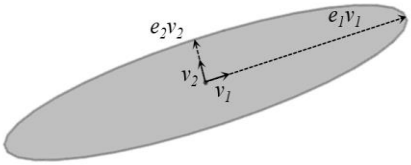


Fig. 8: The Eigen values and Eigen vectors of a Gaussian.

3.2 Training the TP-GMM

A GMM for Frame *j* is trained using an initialisation process then the Expectation Maximisation (EM) algorithm (i.e., E-step and M-step).

Initialisation: Time-based initialisation is performed, and the path points of all demonstrations are divided into *K* groups using the *k*-means algorithm. Gaussian *k* ($k \in \mathbb{N}\ \{1, \ldots, K\}$) is calculated to model the k^{th} group of path points. After initialisation, the E-step and the M-step in the EM algorithm are repeated iteratively until algorithm convergence.

E-step: Let $p_{(t,j)}$ be the series of points $[p_{(t,j)}^{(1)}, \ldots, p_{(t,j)}^{(n)}]$ for $n \in \mathbb{N}\ \{1, \ldots, N\}$. The posterior probability $\gamma_{(t)}^{k}$ of each path point $p_{(t,j)}$ given Gaussian *k* can be calculated as the following equation:

$$\gamma_{(t)}^{k} = \frac{\omega_{(k)} \prod_{j=1}^{J} N(p_{(t,j)}|\mu_{(k,j)},\Sigma_{(k,j)})}{\sum_{k=1}^{K} \omega_{(k)} \prod_{j=1}^{J} N(p_{(t,j)}|\mu_{(k,j)},\Sigma_{(k,j)})} \tag{6}$$

where ω_k is the weight of Gaussian *k*; $N(p_{(t,j)}^{(n)}|\mu_{(k,j)}, \Sigma_{(k,j)})$ is the probability density function of Gaussian *k*, such that:

$$N\left(p_{(t,j)}\middle|\mu_{(k,j)}, \Sigma_{(k,j)}\right) = \frac{1}{\sqrt{2\pi|\Sigma_{(k,j)}|^2}} e^{-\frac{\left(p_{(t,j)}-\mu_{(k,j)}\right)^T \Sigma_{(k,j)}^{-1} \left(p_{(t,j)}-\mu_{(k,j)}\right)}{2}} \tag{7}$$

The posterior probability $\gamma_{(t,n,j)}^{(k)}$ describes the contribution of Gaussian *k* when sampling the path point $p_{(t,j)}^{(n)}$. This value is used to update the parameters of each Gaussian in the next step.

M-step: In the separate GMM for each frame *j*, updated parameters $\{\mu_{(k,j)}, \Sigma_{(k,j)}, w_{(k)}\}$ are calculated for Gaussian *k* below:

$$\mu_{(k,j)} = \frac{\sum_{t=1}^{T} \gamma_{(t)}^{k} p_{(t,j)}}{\sum_{t=1}^{T} \gamma_{(t)}^{k}} \tag{8}$$

$$\Sigma_{(k,j)} = \frac{\sum_{t=1}^{T} \gamma_{(t)}^{k} (p_{(t,j)}-\mu_{(k,j)})(p_{(t,j)}-\mu_{(k,j)})^T}{\sum_{t=1}^{T} \gamma_{(t)}^{k}} \tag{9}$$

$$\omega_{(k)} = \frac{1}{T} \sum_{t=1}^{T} \gamma_{(t)}^{k} \tag{10}$$

The EM algorithm converges on Gaussian parameters when fitting of the Gaussian distributions on path points is maximised. However, as can be noticed from the equations, the algorithm doesn't differentiate between path points that belong to different demonstrations. That is, path points from all demonstrations are regarded equally, which leads to these issues: (1) There is no guarantee that a Gaussian describes points from different demonstrations evenly; (2) There is also no guarantee that a Gaussian does not end up describing a group of points belonging to the same demonstration, i.e., overfitting to a demonstration. This will be solved in the improved TP-GMM/R algorithm presented in Section 4.

3.3 Reproducing Paths Using TP-GMR

TP-GMR is a regression process to reproduce a path given new task parameters. The time step t ($t \in \mathbb{N}\{1, \ldots, T\}$) is the sequence order of a point along the path; therefore, it is taken as the input for the regression process. GMR exploits the joint probability between the observed t and the unobserved $[x,y]$ of each point. For each Frame j, given a GMM made of K Gaussians and for each given time step t, a new Gaussian $\mathcal{N}(\mu_{GMR(t,j)}, \Sigma_{GMR\,(t,j))})$ is calculated at every time step t in the GMR process (illustrated in Fig. 9).

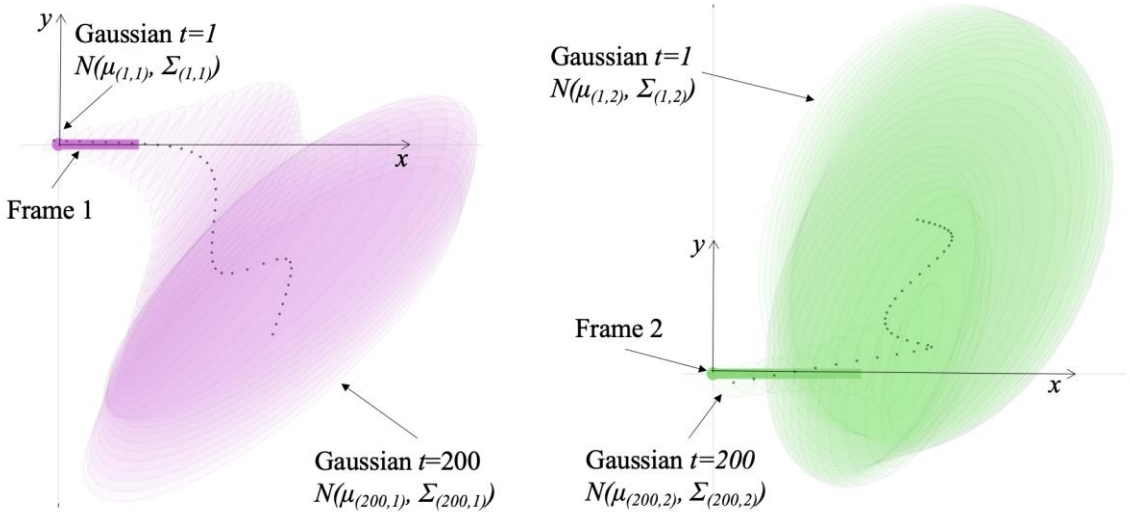


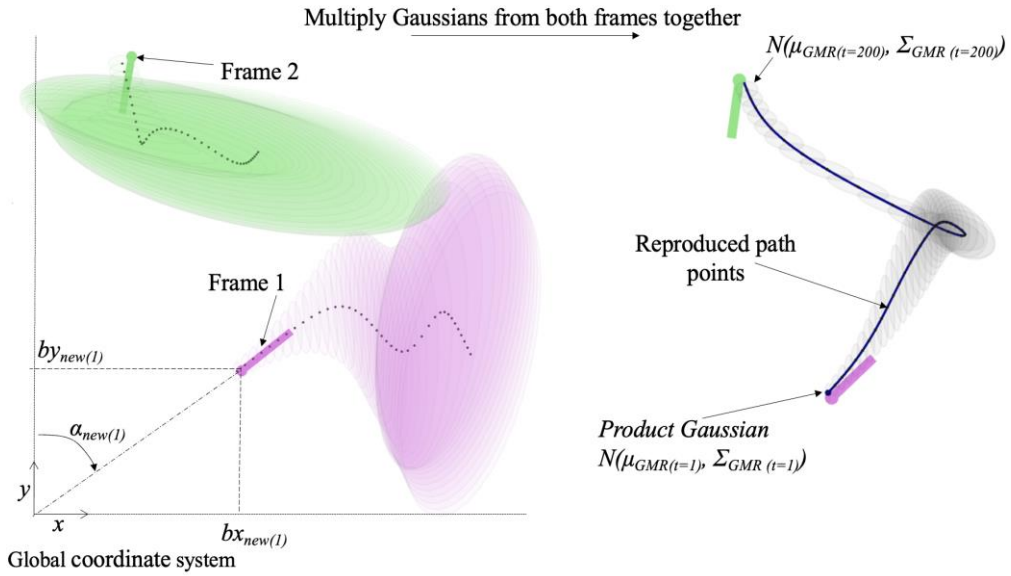
Fig. 9: Given the resultant GMM, K Gaussians from Fig. 7 are regressed to obtain T Gaussians for each time step t. This is the process of TP-GMR.

In a new environment, the position $b_{new\,(j)}=[bx, by]$ and Rotation matrix $A_{new\,(j)}$ of Angle $\alpha_{new(j)}$ are detected for Frame j. For every time step t, the Gaussians of Frame j is converted from the local coordinate system of Frame j to the global coordinate system. Then, a product Gaussian is obtained by combining the Gaussians from different frames using the following equation:

$$N\left(\mu_{GMR\,(t)}, \Sigma_{GMR\,(t)}\right) = \prod_{j=1}^{J} N\left(A_{new\,(j)}\mu_{GMR\,(t,j)} + b_{new\,(j)},\ A_{new\,(j)}\Sigma_{GMR\,(t,j)}\ A_{new\,(j)}{}^{T}\right) \qquad (11)$$

where $b_{new\,(j)}$ is the position vector and $A_{new\,(j)}$ is the rotation matrix for Frame j as detected in the new environmental setup.

The points of a reproduced path are given by the mean values $\mu_{GMR\,(t)}$ (illustrated in Fig. 10). The covariance values of the points, i.e., $\Sigma_{GMR\,(t)}$, reflects how flexible and adjustable the path is at time step *t*, while it still respects the information learnt from demonstrations. Thus, this value has the potential to be utilised to alter the stiffness of a cobot at time step *t* or to vary the position of the path point under other optimisation considerations, but this is beyond the scope of this paper. For more mathematical details about TP-GMR, please refer to Calinon's paper [13].



(a) The frames are in new positions. The regressed Gaussians for each frame are positioned in the space according to the new position and orientation of the frames; (b) The series of product Gaussians $N(\mu_{GMR\,(t)}, \Sigma_{GMR\,(t)})$ for t ($t \in \mathbb{N}$ {1, …, T}). The mean of these Gaussians is equivalent to a reproduced path point. In the example, t=1 and 200 are chosen.

Fig. 10: An example of generating a reproduced path using TP-GMR.

4. Methodology of the Improved TP-GMM/R Algorithm

4.1 Overview

Based on the TP-GMM/R algorithm, an improved TP-GMM/R algorithm is developed. The overall framework entails the following steps: 1) Collecting key information in demonstrations, i.e., task parameters (frames) and paths (refer to Section 3.1). A user provides demonstrations by dragging the end effector of a cobot to create some trajectories from the start position to the end position of a task. 2) Training TP-GMM based on demonstrations (refer to Section 3.2). 3) Identifying orientation-less frames (OLF) and oriented frames (OF). 4) Calculating *r*Gaussians (ring Gaussians) for the OLFs. 5) Upon being given new data for task parameters, converting the *r*Gaussians of OLFs to Gaussians. 6) Performing the TP-GMR algorithm to reproduce a path in a new scenario (refer to Section 3.3). The improved TP-GMM/R algorithm over the "conventional" ones lies in Steps 3, 4, and 5, which are further elaborated in the following. Fig. 11 illustrates the complete steps of the algorithms. As aforementioned, to simplify representations and formulas, in this research, demonstrations are presented in a two-

dimensional Euclidean space for better explanation and illustration. The developed algorithms can be naturally extended to the three-dimensional Euclidean space.

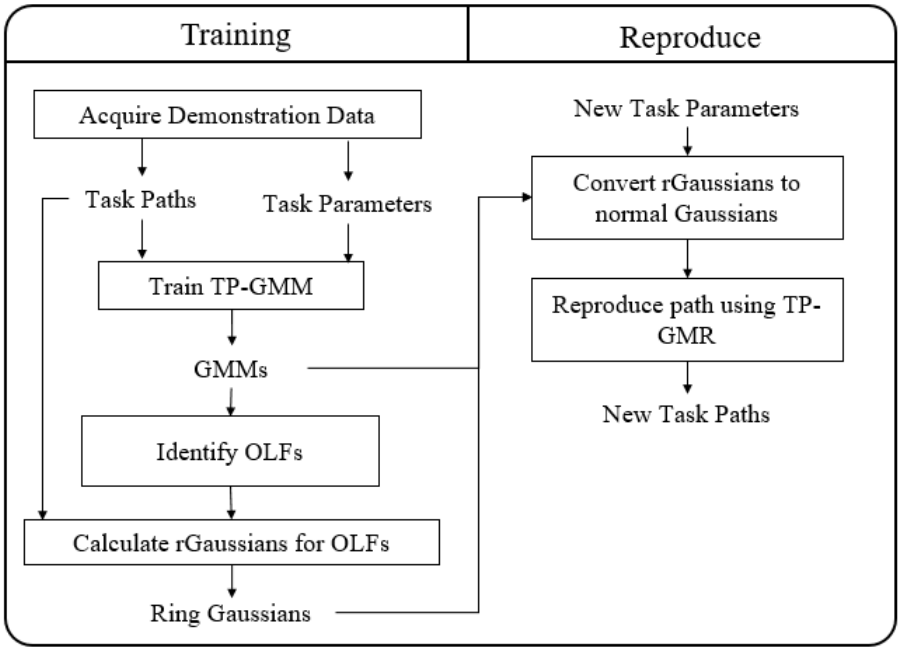


Fig. 11: The overall framework of the improved TP-GMM/R algorithm in this research.

4.2 Identifying OFs and OLFs

TP-GMM models demonstrations with respect to frames, which are defined by their positions and orientations. A frame could be an OLF or an OF. To illustrate what an OLF is, it is useful to elaborate what an OF is using an example presented in Fig. 12, in which a suction cup is used to pick up a ball and place it into a cylindrical hole along a top-down direction. The frame associated with the end position of placing the ball is an OF, which is associated with the rigid body of the cylindrical hole. The geometric constraint of the direction restricts a portion of each demonstration with respect to it. Using TP-GMM, the restricted portion of a demonstration is captured as a Gaussian to identify the associated frame is either an OLF or an OF.

Before the identification of frames, the following should be conducted to pre-processing Gaussians.

Pre-processor: The Gaussian should encode points from all demonstrations

TP-GMM is designed to model patterns and trends in all demonstrations. That is, each resultant Gaussian needs to provide information about the patterns and trends of all demonstrations. However, one of the limitations of TP-GMM is that it does not have knowledge of which points belong to which demonstration. That is, all points are regarded equally during TP-GMM training so that the algorithm might generate Gaussians that are biased towards certain demonstrations over others. In some cases, it might even make a Gaussian to over-fit a demonstration, as depicted in Fig. 13.

As described in Section 3, TP-GMM is trained using the EM algorithm. In every E-step, the likelihood $l_{(tk)}^{(n)}$ of each point $p_{(t)}^{(n)}$, from Demonstration n at time step t belonging to a Gaussian $\mathcal{N}(\mu_{(k)},$

$\Sigma_{(k)}$) is calculated. The calculation is conducted independently for each point, which means that the EM algorithm does not account for which demonstration that point belongs to. This potentially allows Gaussians to over-fit a particular demonstration. Based on this observation, it is essential to introduce adjustments to the EM training algorithm to ensure each obtained Gaussian models points from all demonstrations equally, to avoid such anomalies.

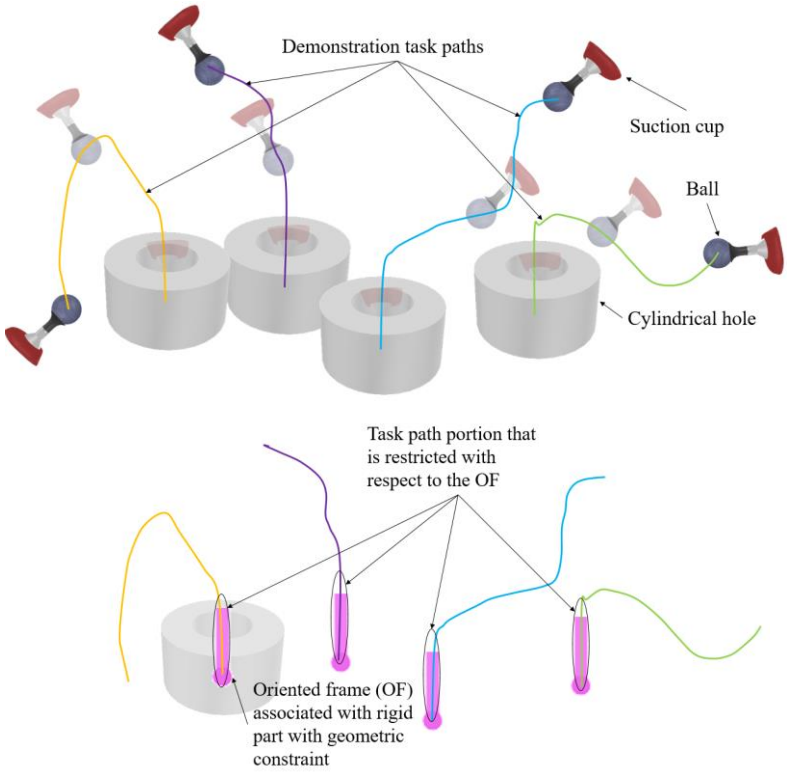
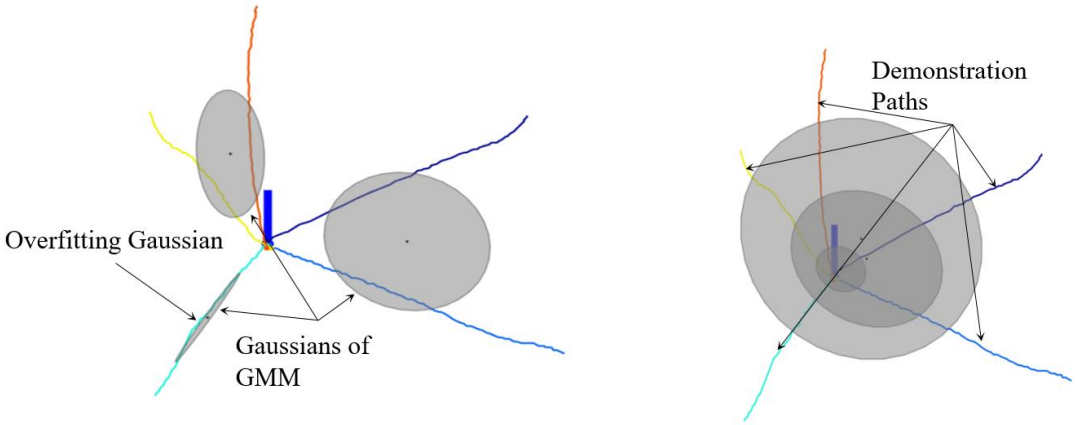


Fig. 12: An example of a task involving an OF.



(a) Result of TP-GMM depicting an anomaly Gaussian over-fitting one demonstration

(b) Result of an adjusted TP-GMM in which over-fitting is avoided.

Fig. 13: Anomaly Gaussian and the adjusted TP-GMM.

For time step t and Gaussian k, $l_{(tk)}^{(n)}$ are averaged over N demonstrations giving $L_{(tk)}$ is defined below:

$$L_{(tk)} = \frac{1}{N}\sum_{n=1}^{N} l_{(tk)}^{(n)} \quad (12)$$

This step associates the likelihoods of the points of the time step *t* together. However, in reality, point $p_{(t)}^{(n)}$ of Demonstration *n* might not be associated with point $p_{(t)}^{(n+1)}$ of Demonstration *n+1*. Demonstrations might have slightly varying lengths and velocities, thus some points from different demonstrations that are closer in the *x-y* space might have slightly different time steps *t*.

Therefore, the values of $L_{(tk)}$ are "blended" across *t* by using weighted average. For each Gaussian *k*, a *kernel* of size *2S+1* is convolved over vector $L_{(tk)}$ across *t*. The vector $L_{(tk)}\ \forall\ t \in \mathbb{N}(1, \dots, T)$ is padded with *S* zeros on each side to allow convolving the filter to the extremities. The information is given below:

$$kernel = [f_1, \dots, f_{S+1}, \dots, f_{2S}, f_{2S+1}]\ where\ f_{S+1-m} = f_{S+1+m} = 1 - \frac{m}{S+1}\ for\ m \in \mathbb{N}\{0, \dots, S\} \quad (13)$$

For example, if *S*=3, *kernel* = [0.25, 0.5, 0.75, 1, 0.75, 0.5, 0.25]. Not using the kernel is equivalent to setting *S*=0.

$$L_{(tk)} = \frac{1}{1+S}\left(\sum_{index=1}^{2S+1} kernel_{index}\ L_{(t-S-1+index,k)}\right) \quad (14)$$

Furthermore, the value of likelihood $l_{(tk)}^{(n)}$ is adjusted by bringing it closer to the average $L_{(tk)}$ by a scale γ (Equation (14)). If γ is set to 1, then values $l_{(tk)}^{(n)}$ (for $n \in \mathbb{N}\{1, \dots, N\}$) would be equal to $L_{(tk)}$. If γ is set to 0, the values $l_{(tk)}^{(n)}$ would be left unchanged. Therefore, an intermediate value of γ, i.e., $\gamma = 0.5$, prevents over-fitting without enforcing rigid constraints.

$$l_{(tk)}^{(n)} = l_{(tk)}^{(n)} + \gamma(l_{(tk)}^{(n)} - L_{(tk)}) \quad (15)$$

Thus, the resultant Gaussians have a tendency to model all demonstrations evenly, instead of over-fitting towards one particular demonstration.

After the pre-processing, an OF can be identified using the following two criteria.

Criterion 1: The Gaussian should be narrow

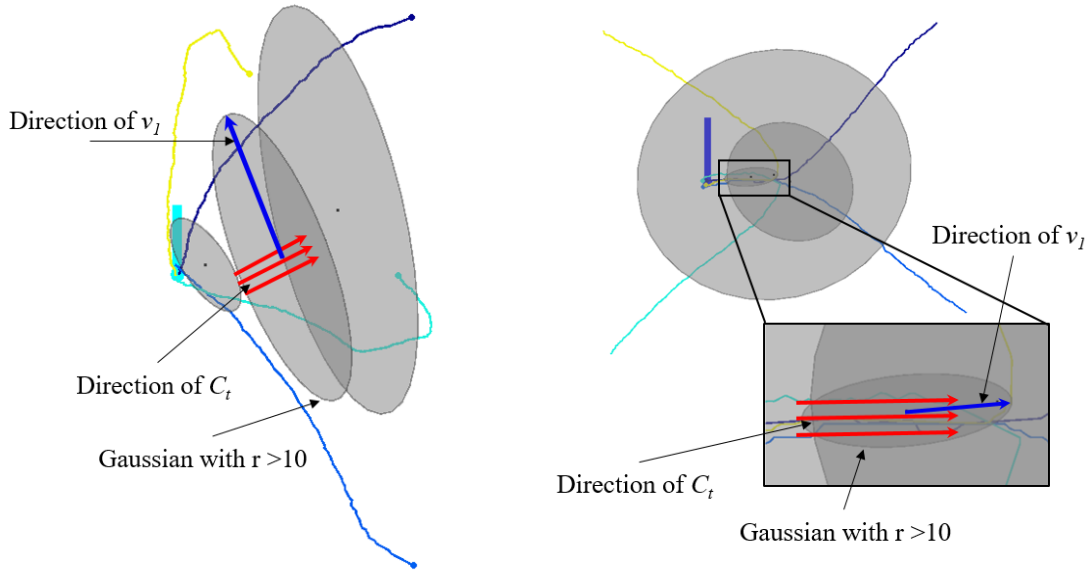
In a Gaussian, assuming Eigen value e_1 is greater than Eigen value e_2, the greater the ratio $r = e_1/e_2$, the more stretched the Gaussian is (Fig. 8). A Gaussian is considered narrow if *r* is greater than a threshold (trials show the 10 is appropriate for the threshold).

A narrow Gaussian indicates that the portions of demonstrations that it encodes are in restricted or limited variability with respect to its corresponding frame. This criterion, when combined with the following criteria, helps identify whether the orientation of that frame affects demonstrations, i.e. if the frame is an OF.

Criterion 2: The Gaussian should stretch along the direction of demonstration paths

This criterion ensures that the Gaussian is not stretched over demonstration paths that are widely apart such as illustrated in Fig. 14(a). Instead, the direction of change of time step *t* (signifying the

direction of demonstrations) should be parallel to the long axis of the Gaussian, such as illustrated in Fig. 14(b).



(a) An example of a narrow Gaussian that does not belong to an OF. The diagram shows that the direction of the change of time step *t* is not parallel to that of the Gaussian's long axis

(b) An example of a narrow Gaussian that belongs to an OF. The diagram shows that the direction of the change of time *t* is almost parallel to that of the Gaussian's long axis

Fig. 14: An example illustrating Criterion 2.

The direction of change of time step *t* is given by the vector $C_t = [C_{xt}, C_{yt}]$, where C_{xt} and C_{yt} are components from the covariance matrix of the Gaussian. C_{xt} describes the effect of the *x* position of a point on its time step *t*. The change in time step *t* signifies the flow on a demonstration, since the points increase time step *t* incrementally along the demonstration. That is, the higher C_{xt} value is, the more the change in the *x* coordinate affects the change in time step *t*. That means that demonstrations are more tending to be parallel to the *x* axis. Similarly, the same applies for C_{yt}. Therefore, the vector $C_t = [C_{xt}, C_{yt}]$ is indicative of the direction of the flow of demonstrations in the *x*-*y* space.

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

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

4.3 Calculating *r*Gaussians

No changes are made to the trained TP-GMM if frames are OFs. However, for OLFs, new probability distributions are generated to model the demonstrations. The points of demonstrations around OLFs are modelled in a new probabilistic distribution, as opposed to the "normal" Gaussian used in the "conventional" TP-GMM/R algorithm. Since the frame is orientation-less, a given path point is equally likely to occur around the frame. Therefore, a suitable Gaussian to describe the distribution

of points around an OLF would be an *r*Gaussian, which is described as a Gaussian that has been spanned around the OLF forming a ring (illustrated in Fig. 15).

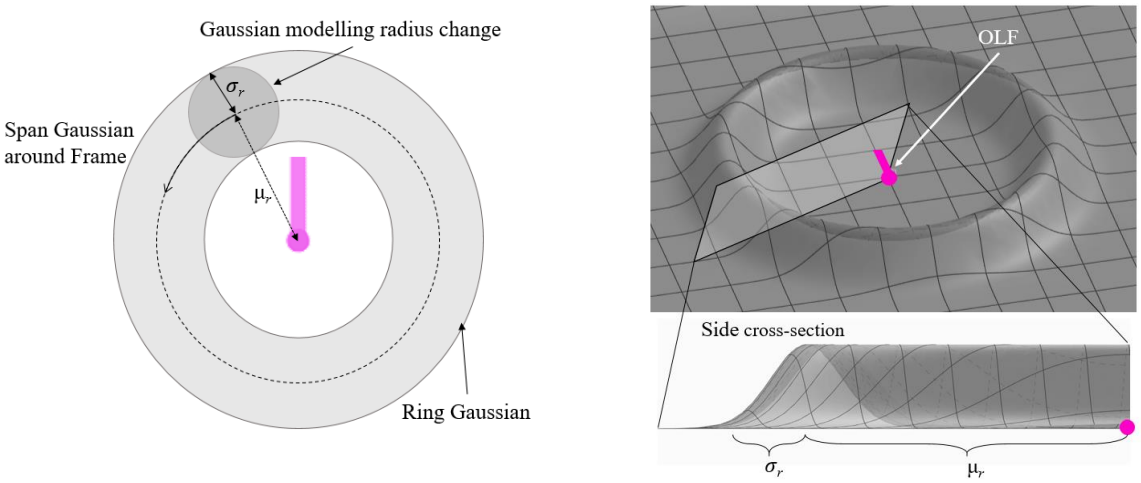


Fig. 15: The *r*Gaussian modelling around an OLF.

An *r*Gaussian is computed below. For every time step *t*, the probability of a path point with respect to the frame is described below:

$$Pr(x,y) = \frac{1}{\sqrt{2\pi{\sigma_r}^2}} e^{-\frac{\left(\sqrt{x^2+y^2}-\mu r\right)^2}{2\sigma {r}^2}} \tag{16}$$

where [*x*,*y*] are the 2D coordinates of the path point relative to the frame, μ_r is the mean distance between the point and the frame, and σ_r is the standard deviation of the distance between the point and the frame.

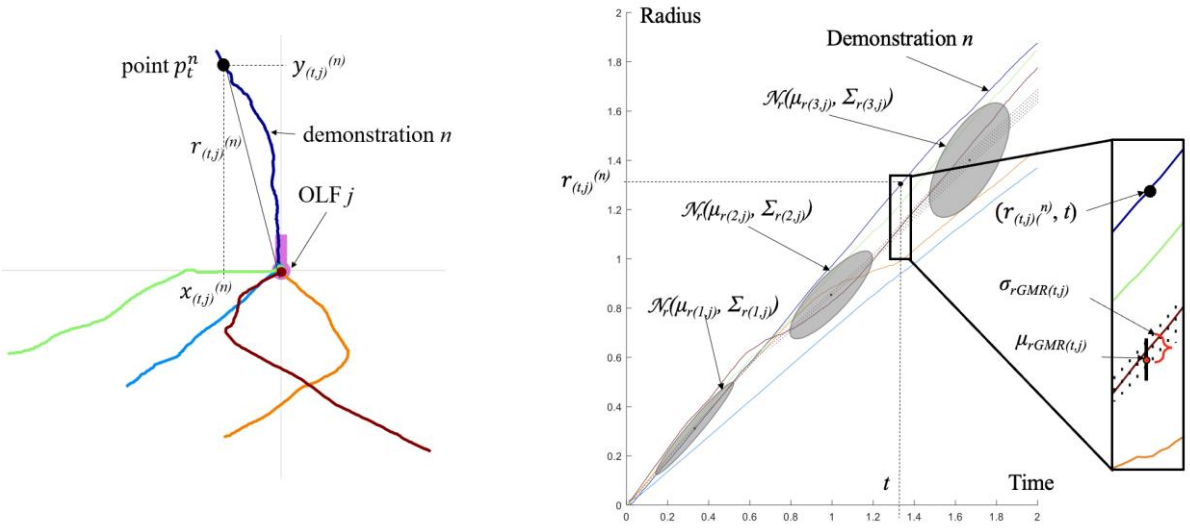
To obtain μ_r and σ_r of every time step *t*, a TP-GMM that models radius *r* and time steps *t* is trained. For example, take OLF *j* in Fig. 16(a) and five demonstration paths (*N*=5) around it. As previously mentioned, each demonstration constitutes of *T* path points represented in time step *t* and the *x*-*y* position coordinates with respect to Frame *j*. To generate the parameters for the *r*Gaussian for Frame *j* at time step *t*, the following steps are followed:

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

$$r_{(t,j)}^{(n)} = \sqrt{\left(x_{(t,j)}^{(n)}\right)^2 + \left(y_{(t,j)}^{(n)}\right)^2} \tag{17}$$

2) A GMM is trained on the 2D data $[t, r_{(t,j)}^{(n)}]$ for all $t \in \mathbb{N}\{1, \ldots, T\}$ and $n \in \mathbb{N}\{1, \ldots, N\}$. This means that *K* Gaussians are fitted to describe the change in the distance between OLF *j* and the path points around it. Gaussian *k* is identified as $\mathcal{N}_r(\mu_{r(k,j)}, \Sigma_{r(k,j)})$. For example, in Fig. 16(b), *K*=3.

3) Gaussian regression is performed to obtain a radius mean μ_rGMR*(t,j)* and standard deviation σ_rGMR*(t,j)* at every time step *t*.



(a) An OLF *j* and five demonstration paths. Each point *p* on a path has [*x*,*y*] with respect to Frame *j*. Moreover, it has a time step *t* depending on its sequence order in its path. The radius *r* is calculated as the distance between the frame and the point; (b) Given radius values of all points from the demonstrations, a GMM is fitted to the radius-time data. Then, regression is performed to obtain a mean μ_rGMR(*t*,*j*) and a standard deviation σ_rGMR(*t*,*j*) for each time *t*.

Fig. 16: Obtaining parameters of a *r*Gaussian.

Finally, the resultant data, average μ_rGMR*(t,j)* and standard deviation σ_rGMR*(t,j)* for every time step *t*, is used in the adjusted TP-GMR in Section 3.3 to reproduce paths in new settings.

4.5 Converting *r*Gaussians to Gaussians

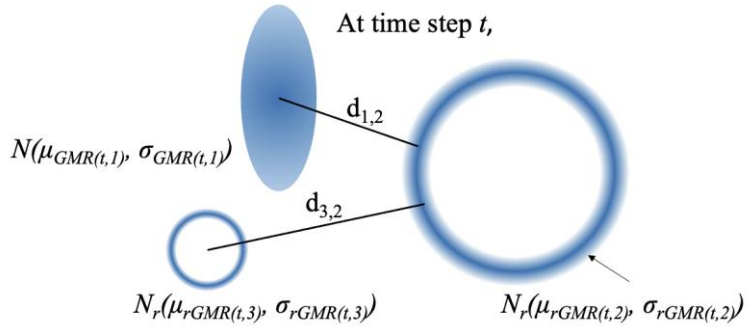
Given new positions of frames, paths should be reproduced using TP-GMR. As a part of TP-GMR, a weighted sum is performed of Gaussians from different frames (refer to Section 3.1 for more details). This step can only be performed on "normal" Gaussians due to their mathematical formulation. Gaussians have a scalar *x*-*y* position as their mean, while the mean of the *r*Gaussian is a function of *x*-*y* coordinates. Therefore, *r*Gaussians of OLFs are converted to Gaussians before performing TP-GMR.

Given an OLF, for time step *t*, the goal is to identify the closest point $p_{closest}$ on its *r*Gaussian $\mathcal{N}_r(\mu_{rGMR(t,j)}, \sigma_{rGMR(t,j)})$ to the Gaussians of the other frames at time step *t*. Finding $p_{closest}$ is essential since when the Gaussians of different frames are closer to each other, the generated path is more likely to adhere to the demonstrations provided. The new Gaussian of Frame *j* at time *t* is going to have its mean $\mu_{GMR(t,j)}=(\mu x_{GMR}, \mu y_{GMR})_{(t,j)}$ at $p_{closest}=[x,y]$ and its covariance $\Sigma_{GMR(t,j)}$ equal to $\sigma_{rGMR(t,j)}$. The Gaussians of other frames could be Gaussians or *r*Gaussians (Fig. 17(a)). For example, for a given time step *t* and three frames, each frame has a Gaussian to describe the distribution of path points at time *t* with respect to the frame. Assume Frame 1 is oriented, i.e., having a Gaussian, while Frames 2 and 3 are orientation-less, i.e., their Gaussians are ring shaped. To convert Frame 2's *r*Gaussian, $\mathcal{N}_r(\mu_{rGMR(t,2)}, \sigma_{rGMR(t,2)})$, to a Gaussian, $\mathcal{N}(\mu_{GMR(t,2)}, \Sigma_{GMR(t,2)})$, the distance $d_{1,2}$ between Frame 2 and oriented Frame 1's Gaussian's

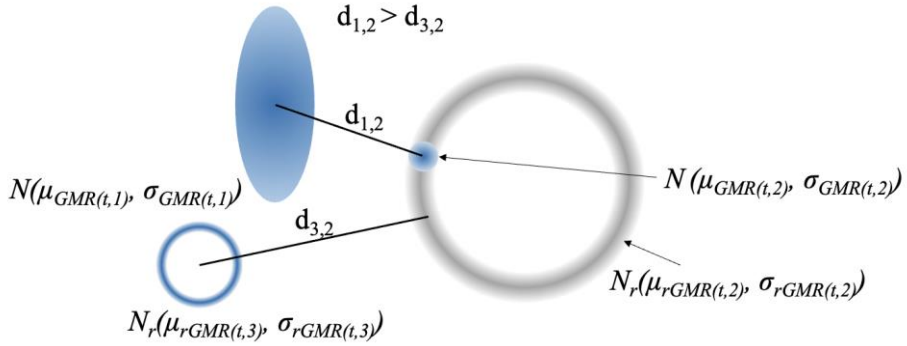
centre μ_t^1 is calculated. Then, the distance $d_{3,2}$ between Frame 2 and orientation-less Frame 3 is calculated. Assuming that $d_{1,2}$ is less than $d_{3,2}$, the closest point $p_{closest}$ on the *r*Gaussian $\mathcal{N}_r(\mu_{rGMR(t,2)}, \sigma_{rGMR(t,2)})$ to Gaussian $\mathcal{N}(\mu_{GMR(t,1)}, \Sigma_{GMR(t,1)})$ is identified. Finally, a Gaussian $\mathcal{N}(\mu_{GMR(t,2)}, \Sigma_{GMR(t,2)})$ is formed with the coordinates of point $p_{closest}$ as mean $\mathcal{N}\{\mu_{GMR(t,2}\}$ and standard deviation $\mathcal{N}(\Sigma_{GMR(t,2)})$ equal to $\mathcal{N}_r(\sigma_{rGMR(t,2)})$ (Fig. 17(b)). Once this process is repeated for all OLFs, TP-GMR will be performed using the converted Gaussians. Table 1 provides the pseudo code for the mentioned process.

Table 1: Algorithm for converting ring Gaussians to normal Gaussians

Inputs
New positions $b_{(j)}^{new}=[bx,by]_{(j)}^{new}$ and orientations $A_{(j)}^{new}$ of Frame $j \in \mathbb{N}\ \{1, \ldots, J\}$
Modelled *r*Gaussians $\mathcal{N}_r(\mu_{rGMR(t,j)}, \sigma_{rGMR(t,j)})$ of Frame j if j is orientation-less
Modelled Gaussians $\mathcal{N}(\mu_{GMR(t,j)}, \Sigma_{GMR(t,j)})$ of Frame j if j is orientated
Algorithm
For every orientation-less Frame $j \in \mathbb{N}\ \{1, \ldots, J\}$
  For every time step $t \in \mathbb{N}\ \{1, \ldots, T\}$
    For every Frame $j_{other} \in \mathbb{N}\ \{1, \ldots, J\}-\{j\}$
      If frame j_{other} is orientation-less
        Calculate distance $d_{jother,\,j}$ between frame positions $b_{(jother)}^{new}$ and $b_{(j)}^{new}$
      Else if frame j_{other} is oriented
      Calculate distance $d_{jother,\,j}$ between frame position $b_{(j)}^{new}$ and Gaussian mean $\mu_{GMR(t,j)}$
      End
    End
    Let j_{min} be the frame for which $d_{jother,\,j}$ is the least
    If Frame j_{min} is orientation-less
      Let point $p_{closest}=[x,y]$ be point on $\mathcal{N}_r(\mu_{rGMR(t,j)}, \sigma_{rGMR(t,j)})$ closest to Frame j_{min}
    Else if Frame j_{min} is oriented
      Let point $p_{closest}=[x,y]$ be point on $\mathcal{N}_r(\mu_{rGMR(t,j)}, \sigma_{rGMR(t,j)})$ closest to Gaussian mean $\mu_{GMR(t,jmin)}$
    End
    Let $\mathcal{N}(\mu_{GMR(t,j)}, \Sigma_{GMR(t,j)})$ be the Gaussian for Frame j at time step t
    Set Gaussian mean $\mu_{GMR(t,j)}=[\mu x_{GMR}, \mu y_{GMR}]_{(t,j)}$ be equal to point $p_{closest}=[x,y]$
    Set Gaussian covariance $\Sigma_{GMR(t,j)}$ be equal to $\mathcal{N}_r(\sigma_{rGMR(t,j)})$
  End
End
Output
Modelled Gaussians $\mathcal{N}(\mu_{GMR(j)}, \Sigma_{GMR(j)})$ of Frame j if Frame j is orientation-less



(a) The Gaussian 1 is found to be closer than Gaussian 3 than to *r*Gaussian 2. The point *P* that is at the shortest distance on *r*Gaussian 2 to Gaussian 1 is identified.



(b) A new Gaussian is formed with its centre being the point *P*. Its standard deviation is similar to that of the *r*Gaussian.

Fig. 17: Transforming an *r*Gaussian of Frame 2 to a normal Gaussian before performing TP-GMR.

5. Results and Discussion

5.1 Assessing the Performance of the OLF Identifier

The performance of the orientation-less frame (OLF) identifier depends on two main factors: 1) the quality and variability of demonstrations provided, and 2) the ratio between *K*, the number of Gaussians in the GMM, and the length of the orientation-consistent path. An orientation-consistent path is the portion of demonstration paths that is dependent on the frame's orientation (described further in Section 3.1). To assess the performance of the OLF identifier, its F1-score is calculated as the following parameters are varied:

- Number of Gaussians (*K*): (3, 4)
- Number of demonstrations used in training (*N*): (4, 5, 6)
- Length of the orientation-consistent path (*L*): (0, 10, 15, 20, 25)% of the average distance between the frames

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

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{18}$$

where,

$$Precision = \frac{nb\ of\ OLFs\ identified\ as\ OLFs}{nb\ of\ frames\ identified\ as\ OLFs} \tag{19}$$

$$Recall = \frac{nb\ of\ OLFs\ identified\ as\ OLFs}{nb\ of\ OLFs} \tag{20}$$

Table 1: F1 scores for 30 different runs of the OLF identifier with varied parameters.

	N=4, *K*=3	*N*=5, *K*=3	*N*=6, *K*=3	*N*=4, *K*=4	*N*=5, *K*=4	*N*=6, *K*=4
L=0%	1	1	1	1	1	1
L=10%	0.67	0.67	1	1	0.67	1
L=15%	1	1	1	1	1	1
L=20%	1	0.67	1	1	1	1
L=25%	1	1	1	1	1	1

In Table 1, an F1 value of 1 indicated than Frame 1 was correctly identified as an OF and Frame 2 was correctly identified as OF=LF (except when *L*=0%, both frames are OLFs). The results show a correlation between the three different parameters and the effectiveness of the OLF identifier. Based on these correlations, the following recommendations are provided for a user to choose parameters that increase the success chances of the OLF identification:

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

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

Table 2: F1 scores for 30 different runs of the OLF identification algorithm with varied parameters.

	N=4, K=3	N=5, K=3	N=6, K=3	N=4, K=4	N=5, K=4	N=6, K=4
L=0%	Inf	Inf	Inf	Inf	Inf	Inf
L=10%	Inf	1	1	Inf	1	Inf
L=15%	1	1	1	1	1	1
L=20%	1	1	1	1	1	1
L=25%	Inf	1	1	Inf	1	1

5.2 Qualitative Results on Synthetic Data

The improved TP-GMM/R algorithm (ours) was tested for several cases of synthesised data that is similar to the standard data used to test the conventional TP-GMM algorithm devised by Calinon [13] (also presented in Section 3). The task is for a point to move from Frame 1 (pink) to Frame 2 (green), under varying conditions. The performance of the improved TP-GMM/R on the task was evaluated when:

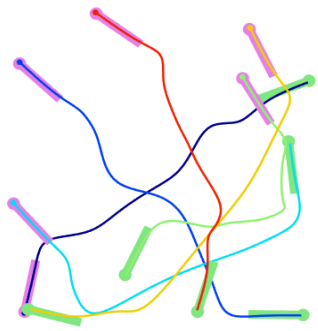
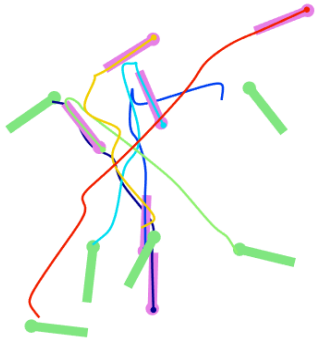
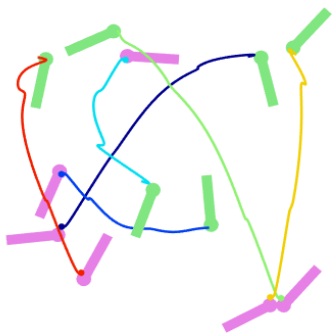
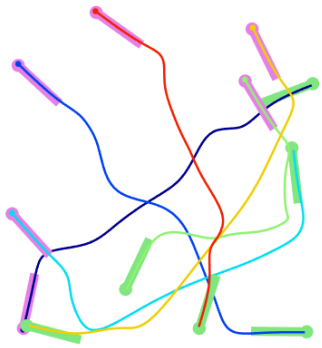
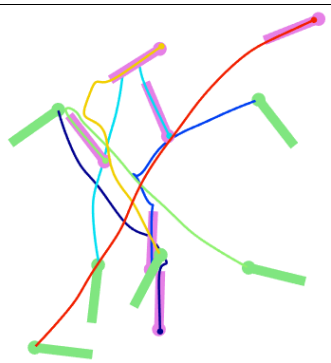
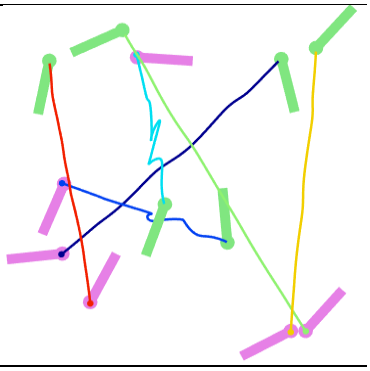
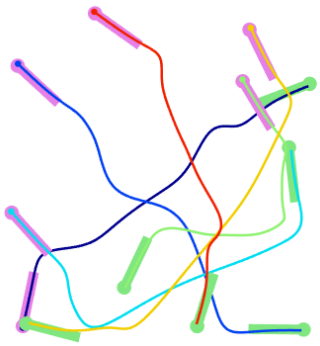
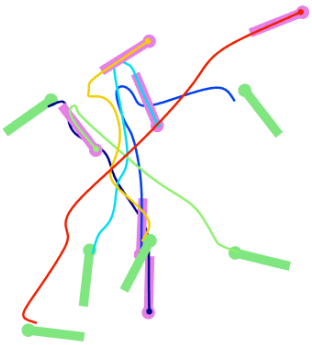
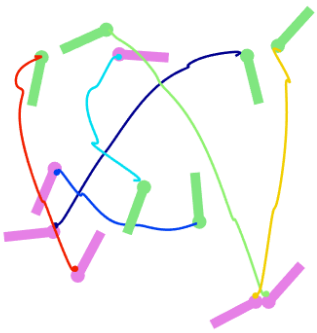
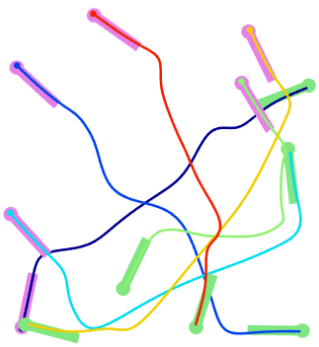
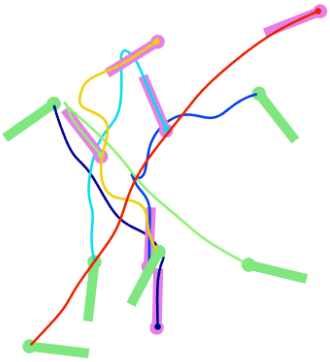
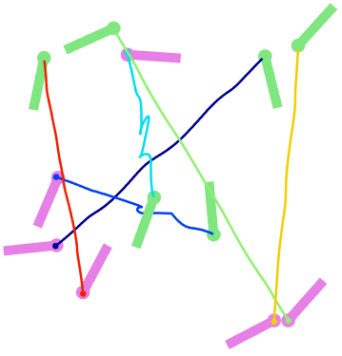
- Task 1: Both frames are oriented (OFs). Therefore, the path has to pass through Frame 1 and Frame 2's tips as seen in Fig. 18(a). The training parameters were: N=4, K=3. This is analogous to a peg-in-hole task.
- Task 2: Frame 1 is oriented (OF) while Frame 2 is orientation-less (OLF). Therefore, the path has to pass through Frame 1's tips but can approach Frame 2 from any direction as seen in Fig. 18(b). The training parameters were: N=6, K=4.
- Task 3: Both frames are orientation-less (OLFs). Therefore, the path can approach Frame 1 and Frame 2 from any direction as seen in Fig. 18(c). The training parameters were: N=6, K=4. This is analogous to a pick-and-place task.

(a) Task 1 (b) Task 2 (c) Task 3

Fig. 18: Demonstrations (ground truth) for training the conventional and improved algorithms.

Table 3 shows the results of the path reproduction on the three tasks using the conventional TP-GMM/R and ours, with and without kernel convolution.

Table 3: Reproduced paths of the three tasks using both algorithms with/without overfitting kernel convolution.

Task 1	Task 2	Task 3
Conventional TP-GMM/R		
		
Improved TP-GMM/R		
		
Conventional TP-GMM/R without kernel		
		
Improved TP-GMM/R without kernel		
		

For Task 1, the OLF identifier was successful in identifying the frames as OFs. Therefore, the performances of the conventional and improved algorithms are identical, since both algorithms run identically when the frames are OFs. The purpose of task 1 was only to show that the performances of both algorithms are identical when all frames are OFs.

For Task 2, it can be noticed that the reproduced paths starting from the oriented Frame 1 respected the orientation constraint (passing through the frame's tip) for both algorithms. That means that even when using ours where an OLF is present in conjunction with an OF, the constraints imposed by the OF are not compromised. Moreover, using the conventional algorithm, the reproduced path seems to attempt to approach the orientation-less Frame 2 from a fixed angle. This causes the path to have awkward and unnecessary bends as well as sometimes fail to reach the target Frame 2. Using the improved algorithm, this problem does not exist since the algorithm accounts for the orientation-less nature of Frame 2 and approaches the frame successfully from the any angle.

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

Overall, the improved algorithm provides comparably satisfactory results with OFs and better results with OLFs than the conventional algorithm. Quantitative comparison and assessment are presented in Section 5.3.

5.3 Quantitative Results on Synthetic Data

Furthermore, the performances of both algorithms on Tasks 1, 2 and 3 was quantitatively compared according to multiple metrics, i.e. smoothness, efficiency and reachability.

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

$$\frac{dy_t}{dx_t} = \frac{y_{t+1} - y_{t-1}}{x_{t+1} - x_{t-1}} \tag{21}$$

$$\frac{d\left(\frac{dy_t}{dx_t}\right)}{dt} = \frac{\left(\frac{dy_{t+1}}{dx_{t+1}} - \frac{dy_{t-1}}{dx_{t-1}}\right)}{2} \tag{22}$$

$$edges_{m,t} = \begin{cases} 1\,, & d\left(\frac{dy_t}{dx_t}\right)/dt > 0.5 \;\; for\; reproduction\; path\; m \in \mathbb{N}\{1, \dots, M\} \\ 0\,, & d\left(\frac{dy_t}{dx_t}\right)/dt \leq 0.5 \;\; for\; reproduction\; path\; m \in \mathbb{N}\{1, \dots, M\} \end{cases} \tag{23}$$

$$Smoothness = \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T} edges_{m,t} \quad (24)$$

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

$$sd_m = \sqrt{(y_1 - y_T)^2 + (x_1 - x_T)^2} \quad (25)$$

$$d_m = \sum_{t=1}^{T-1}\sqrt{(y_t - y_{t+1})^2 + (x_t - x_{t+1})^2} \quad (26)$$

$$Efficiency = \frac{1}{M}\sum_{m=1}^{M}\frac{sd_m}{d_m} \quad (27)$$

Reachability. The *reachability* shows the percentage of the path successfully passing in the start and end points. The reason to consider this measurement is that even if a reproduced path is smooth and efficient, if it does not achieve successful reachability, the task fails. Firstly, for each reproduction path m, the distance $d1_m$ between the start frame $[X_1, Y_1]$ and the start point $[x_1, y_1]$ on the path is measured. Similarly, the distance $d2_m$ between the end frame $[X_2, Y_2]$ and the last point $[x_2, y_2]$ on the path is measured. For each value of $d1_m$ or $d2_m$ that is greater than an error tolerance value ε, the *reachability* score is increased by 0.05. The error tolerance is manually chosen and can change depending on the task and the scale/size of the task objects involved. Finally, the *reachability* score is averaged over the total number of reproductions M.

$$d1_m = \sqrt{(y_1 - Y_1)^2 + (x_1 - X_1)^2} \quad (28)$$

$$d2_m = \sqrt{(y_2 - Y_2)^2 + (x_2 - X_2)^2} \quad (29)$$

Starting from *Reach* = 0 and iterating over the values of $m \in \mathbb{N}(1, \dots, M)$:

$$Reach = \begin{cases} Reach, & d1_m > \varepsilon \text{ and } d2_m > \varepsilon \\ Reach + 0.5, & d1_m < \varepsilon \text{ or } d2_m < \varepsilon \\ Reach + 1, & d1_m < \varepsilon \text{ and } d2_m < \varepsilon \end{cases} \quad (30)$$

$$Reach = Reach \times 100 \,/\, M \quad (31)$$

Table 4 shows the results of the above metric when the different versions of the algorithm where run on Tasks 2 and 3.

Table 4: Metric values for test run on Tasks 2 and 3 for the conventional and improved algorithms.

Task/Algorithm		Smoothness	Efficiency	Reachability (%)
Task 1	Conventional/Improved	43.5200	2.2506	99
	Conventional/Improved without kernel	43.8500	2.2368	96
Task 2	Conventional	46.55	1.499	62.5

	Improved	29.95	1.456	100
	Conventional without kernel	45.8300	1.5143	64
	Improved without kernel	36.7800	1.4527	100
Task 3	Conventional	50.76	1.1123	62.5
	Improved	29.78	1.0698	100
	Conventional without kernel	53.6300	1.1207	70
	Improved without kernel	31.8000	1.0628	100

From Table 4, the following can be concluded:

- It shows that the kernel convolution is still necessary for the success of the OLF identifier algorithm as explain in Section 4.2.
- The improved algorithm reproduces smoother and straighter paths since the smoothness score (the average number of sharp turns) is less in the improved algorithm than in the conventional algorithm for both Tasks 2 and 3. Moreover, since the paths by the improved algorithm are straighter, they are also more efficient.
- The improved algorithm provides a 100% success rate of reaching the target start and end frames under the stated error tolerance ε=0.05. The success rate of the conventional algorithm is considerably low when the tasks involve an OLF, i.e., Tasks 2 and 3.

Therefore, it can be concluded that the improved TP-GMM/R algorithm provides better or comparable results to the conventional algorithm.

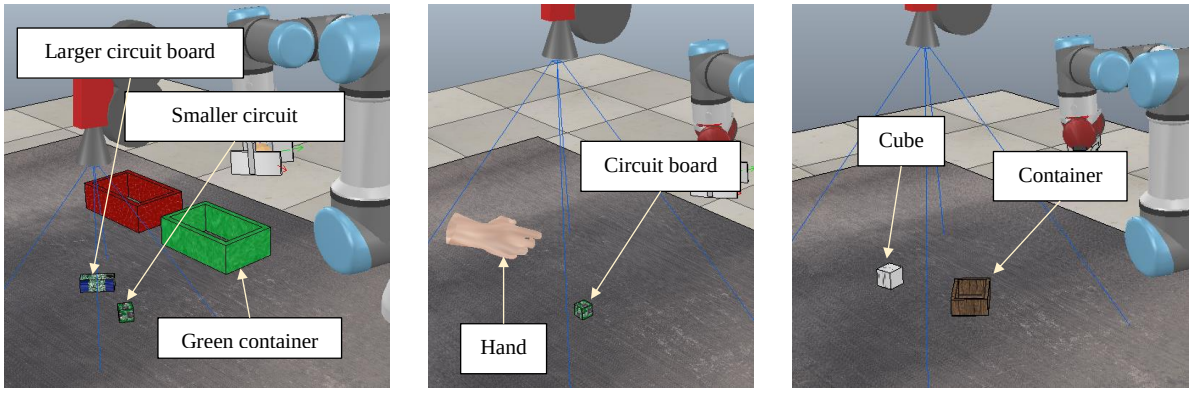
5.4 Simulation Experiments

To assess the results of the improved TP-GMM/R algorithm, the experiment on Tasks 1, 2 and 3 described below are conducted (also illustrated in Fig. 19). The effectiveness of the *r*Gaussian at modelling paths with respect to OLFs is examined. The simulation scenes are built in the CoppeliaSim EDU. The tasks performed are the following:

Task 1 - Sorting: Given two circuit boards and two containers, the task of a cobot is to pick the smaller circuit board and place it in a green container (Fig. 19 (a)). The containers are in fixed positions, outside the camera's view whereas the circuit boards vary positions and orientations. Task 1 shows that ring Gaussians can cater for OF even if they are fixed in positions and out of camera view.

Task 2 - Handover: A cobot needs to pick up an object, in this case a circuit board, and hands it to a human hand (Fig. 19 (b)). Both the hand and the circuit board vary positions and orientations.

Task 3 - Pick-and-place: A cobot is required to pick an object, i.e., a cube, and place it in a container (Fig. 19 (c)). Both the cube and the container vary positions and orientations. This could also be analogous to peg-in-hole or assembly tasks.



(a) Task 1 (b) Task 2 (c) Task 3

Fig. 19: The task simulation scenes of: (a) Task 1, the sorting; (b) Task 2, the handover; and (c) Task 3, the pick-and-place.

Six demonstrations were recorded for each task where the objects are in variable positions. Five of these demonstrations were used for training, and one of them was used for validation. The task parameters, i.e., frames, were detected and identified using our previous work, *i*TP-LfD [19]. The following parameters were set as defaults during training and were not changed between tasks:

- Number of demonstrations $M = 5$
- Number of Gaussian components $K = 4$

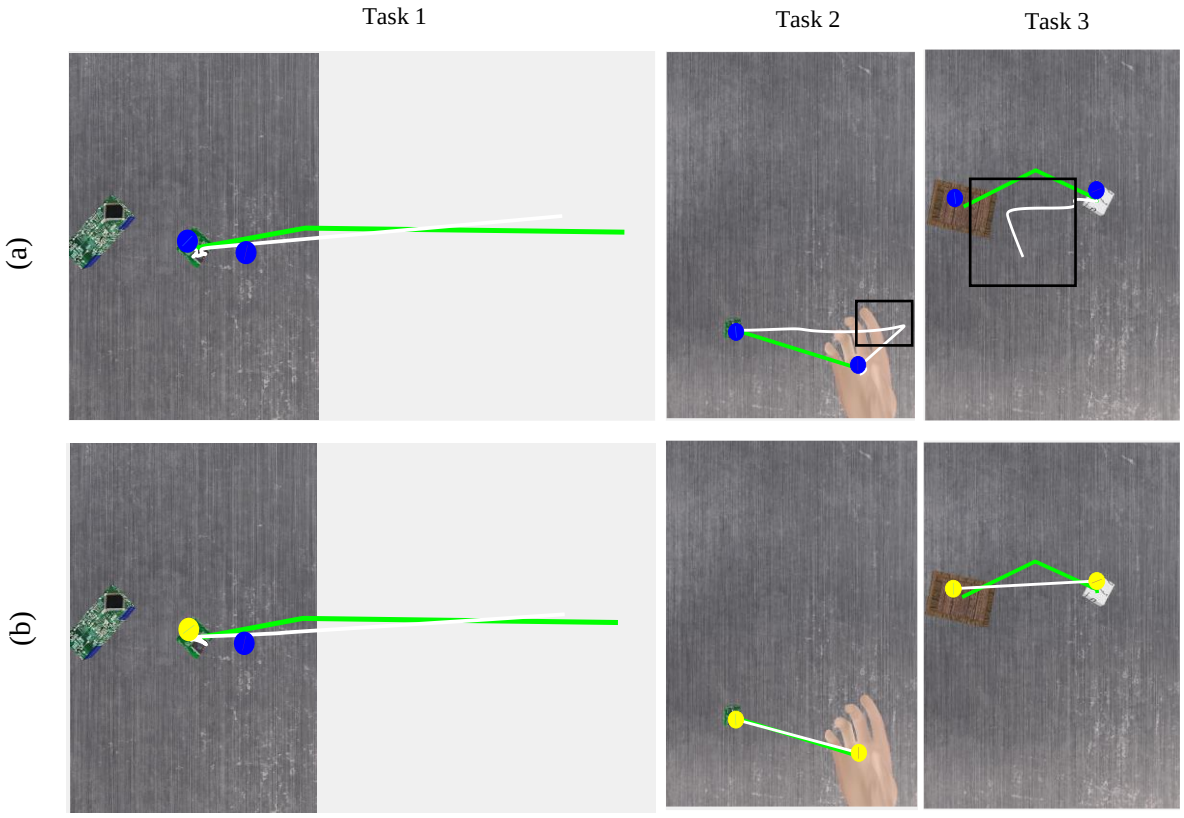


Fig. 20: The reproduced path for Tasks 1, 2 and 3 on the validation image when: (a) relevant frames are used in the conventional TP-GMM algorithm and considered OFs (blue) by default, and (b) relevant frames are used in the improved TP-GMM algorithm and identified to be OLFs (yellow) or OFs (blue).

Fig. 20 shows the results of the algorithms (the conventional or improved TP-GMM/R algorithms) at different stages on the validation demonstration. Row (a) shows the reproduced path (white) compared to the demonstration path (green) when the conventional TP-GMM/R algorithm is used and the relevant frames are identified and considered oriented by default (the reason that the right sides of Task 1 in Fig. 20 are not shown is because the entire space will become tight for the boards to move. The benefit is that the *i*TP-LfD works even if an object of interest is out of view as long as it is fixed in position). The resultant reproduced path fails to accomplish Task 3. Moreover, the reproduced path is inefficient in accomplish Task 2. Row (b) shows the reproduced path (white) compared to the demonstration path (green) when the improved TP-GMM/R algorithm is used and some of the relevant frames are identified to be orientation-less. The reproduced paths all reach their target locations and are smooth.

Table 5: The distances between the reproduced paths and the ground truth (validation demonstration) when all frames are considered OFs or when frames are identified as OFs or OLFs. The results are shown when all lead frames are used in TP-GMR or when only the relevant frames are used.

Task		d_{total}	d_{start}	d_{end}
Task 1	All OFs	0.0127	0.0004	0.0412
	OFs & OLFs	0.0113	0.0024	0.0346
Task 2	All OFs	0.0726	0.0039	0.0086
	OFs & OLFs	0.0097	0.0078	0.0058
Task 3	All OFs	0.0510	0.0048	0.0747
	OFs & OLFs	0.0248	0.0177	0.0121

Table 5 shows the distances between the reproduced path and the ground truth (validation demonstration) in multiple situations:

- d_{total} is the average distance between all the points on the reproduced path and the demonstration.
- d_{start} is the distance between the first point on the reproduced path and the demonstration.
- d_{end} is the distance between the last point on the reproduced path and the demonstration.

Each metric is calculated twice for each task:

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

The distance is highlighted to be:

- Green if the distance of the OFs and OLFs is less than that of the OFs alone. This means that the improved TP-GMM/R algorithm increases the similarity between the demonstration path and the reproduced path.

- Yellow if the distance of the OFs and OLFs is comparable to that of the OFs alone, i.e. with a difference of less than 0.01.
- Red if the distance of the OFs and OLFs is higher than that of the OFs alone.

The results show that using *r*Gaussians and the improved TP-GMM/R algorithm generally improve or maintain the similarity between the demonstration and the reproduced path compared to using the normal Gaussian alone.

6. Conclusions

The conventional TP-GMM/R algorithm intrinsically accounts for the frames' orientations when modelling task paths. However, some frames' orientation is irrelevant to the task paths. Such frames are called orientation-less frames. Using conventional TP-GMM/R with orientation-less frames leads to sub-optimal results since the algorithm doesn't capture the flexibility of the path with respect to the frames' orientations. In fact, the task reproduction will be biased towards what is observed in the demonstrations. Therefore, in this research, a new Gaussian model, i.e., *r*Gaussian, that capture the flexibility of the paths with respect to the frame orientation, is presented. This new model is integrated with the conventional TP-GMM/R algorithm as an improved TP-GMM/R algorithm to overcome overfitting and better serve the purpose of TP-GMM/R. The designed algorithm is tested on a series of simulation tasks of varied complexity and purpose, and it proved to perform better than the conventional TP-GMM/R.

Currently, the improved algorithm is applicable to 2D planar motions. However, it is extendable to 3D by calculating 3D Gaussian mixture models and spherical Gaussians instead of *r*Gaussians. We aim to explore this extension more comprehensively in the future. Moreover, the performance of the algorithm is highly dependent on the quality of the demonstrations provided. The user should ensure that the demonstrations exhibit task variability where applicable to ensure a meaningful and functional learnt task model. In future works, we aim to explore interactive demonstration building, in which an algorithm intelligently suggests demonstrations for the user to perform to maximise variability. Finally, we aim to run a user study to explore how easy and applicable for operators to use the algorithm to program cobots. That is to go in line with the initial motivation of the research project, which is to provide operators with an intuitive programming approach effective in teaching cobots flexible behaviours in manufacturing.

Acknowledgments

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

References

[1] R. Müller, M. Vette and O. Mailahn, "Process-oriented task assignment for assembly processes with human-robot interaction," *Procedia CIRP*, vol. 44, pp. 210--215, 2016.

[2] V. Villani, F. Pini, F. Leali and C. Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248-266, 2018.

[3] F. Vicentini, "Terminology in safety of collaborative robotics," *Robotics and Computer-Integrated Manufacturing*, vol. 63, Article ID 101921, 2020.

[4] S. E. Zaatari, M. Marei, W. D. Li and Z. Usman, "Cobot programming for collaborative industrial tasks: An overview," *Robotics and Autonomous Systems*, vol. 116, pp. 162-180, 2019.

[5] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, 2018.

[6] Z. Zhu, H. Hu and D. Gu, "Robot performing peg-in-hole operations by learning from human demonstration," in *Proceedings of the 2018 10th Computer Science and Electronic Engineering (CEEC)*, 2018.

[7] D. Kent, M. Behrooz and S. Chernova, "Construction of a 3D object recognition and manipulation database from grasp demonstrations," *Autonomous Robots*, vol. 40, no. 1, pp. 175-192, 2016.

[8] L. Schwenkel, M. Guo and M. Burger, "Optimizing sequences of probabilistic manipulation skills learned from demonstration," in *Proceedings of the Conference on Robot Learning, PMLR*, 2020.

[9] K.Kronander, M.Khansari and A.Billard, "Incremental motion learning with locally modulated dynamical systems," *Robotics and Autonomous Systems*, vol. 70, pp. 52-62, 2015.

[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 the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[11] Z. Jia, M. Lin, Z. Chen and S. Jian, "Vision-based robot manipulation learning via human demonstrations," 2020, arXiv:2003.00385.

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

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

[14] S. Calinon, F. D-halluih, E. Sauser, D. Caldwell and A. Billard, "Learning and reproduction of gestures by imitation," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 44-54, 2010.

[15] 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.

[16] S. Hu and K. J. Kuchenbecker, "Hierarchical task-parameterized learning from demonstration for collaborative object movement," *Applied Bionics and Biomechanics*, Article ID 9765383, 2019.

[17] S. Calinon, "Robot learning with task-parameterized generative models," in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2015.

[18] M. Kyrarini, M. A. Haseeb, D. Ristic-Durrant and A. Gräser, "Robot learning of industrial assembly task via human demonstrations," *Autonomous Robots*, vol. 43, p. 239–257, 2019.

[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] S. E. Zaatari, Y. Wang, W. D. Li and Y. Peng, "iTP-LfD: Improved task parametrised learning from demonstration for generic cobot programming," *Robotics and Computer-Integrated Manufacturing*, vol. 69, Article ID 102109, 2021.

[21] K. Fischer, F. Kirstein, L. C. Jensen, N. Kruger, K. Kuklinski, M. V. a. d. Wieschen and T. R. Savarimuthu, "A comparison of types of robot control for programming by demonstration," in *Proceedings of the 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2016.

[22] C. Yang, C. Zeng, P. Liang, Z. Li, R. Li and C.-Y. Su, "Interface design of a physical human–robot interaction system for human impedance adaptive skill transfer," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 329-340, 2018.

[23] U. E. Ogenyi, G. Zhang, C. Yang, Z. Ju and H. Liu, “An intuitive robot learning from human demonstration,” in *International Conference on Intelligent Robotics and Applications*, 2018.

[24] 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.

[25] A. Rogowsk and P. Skrobek, “Object identification for task-oriented communication with industrial robots,” *Sensors*, vol. 20, no. 6, pp. 1773, 2020.

[26] A. Rivera, K. Spasovski, S. Athavale and R. Nunez, “Manufacturing part identification using computer vision and machine learning”. Patent 20190244008, 2018.

[27] C. Paxton, A. Hundt, F. Jonathan, K. Guerin and G. D. Hager, “CoSTAR: Instructing collaborative robots with behavior trees and vision,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

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

[29] Z. Cao, H. Hu, Z. Zhao and Y. Lou, “Robot programming by demonstration with local human correction for assembly,” in *Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019.

[30] C. Willibald, “Development of an interactive robot programming method,” Technische Universität München, Master Dissertation, 2020.

[31] A. Sena, B. Michael and M. Howard, “Improving task-parameterised movement learning generalisation with frame-weighted trajectory generation,” in *Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[32] J. Vidaković, B. Jerbić, B. Šekoranja, M. Švaco and F. Šuligoj, “Learning from demonstration based on a classification of task parameters and trajectory optimization,” *Journal of Intelligent and Robotic Systems,* vol. 99, no. 2, pp. 261-275, 2019.

[33] J. Silverio, S. Calinon, L. Rozo and a. D. G. Caldwell, “Learning task priorities from demonstrations,” *IEEE Transactions on Robotics,* vol. 35, no. 1, pp. 78-94, 2019.

[34] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez and C. Torras, “Learning physical collaborative robot behaviors from human demonstrations,” *IEEE Transactions on Robotics,* vol. 32, no. 3, pp. 513-527, 2016.

[35] S. Calinon, F. Guenter and A. Billard, “On learning, representing and generalizing a task in a humanoid robot,” *IEEE Transactions on Systems, Man and Cybernetics,* vol. 37, no. 2, pp. 286-298, 2007.