TEAM: A Parameter-Free Algorithm to Teach Collaborative Robots Motions from User Demonstrations

Lorenzo Panchetti[1][a], Jianhao Zheng[1][b], Mohamed Bouri[1][c] and Malcolm Mielle[2][d]

[1]*École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*

[2]*Schindler AG, EPFL Lab, Lausanne, Switzerland*

ﬂ

Keywords: Learning from Demonstration, Cobots, Probabilistic Movement Primitives, Industrial Applications.

Abstract: Learning from demonstrations (LfD) enables humans to easily teach collaborative robots (cobots) new motions that can be generalized to new task configurations without retraining. However, state-of-the-art LfD methods require manually tuning intrinsic parameters and have rarely been used in industrial contexts without experts. We propose a parameter-free LfD method based on probabilistic movement primitives, where parameters are determined using Jensen-Shannon divergence and Bayesian optimization, and users do not have to perform manual parameter tuning. The cobot's precision in reproducing learned motions, and its ease of teaching and use by non-expert users are evaluated in two field tests. In the first field test, the cobot works on elevator door maintenance. In the second test, three factory workers teach the cobot tasks useful for their daily workflow. Errors between the cobot and target joint angles are insignificant—at worst 0.28 deg—and the motion is accurately reproduced—GMCC score of 1. Questionnaires completed by the workers highlighted the method's ease of use and the accuracy of the reproduced motion. Public implementation of our method and datasets are made available online.

1 INTRODUCTION

Collaborative robots (cobots) are built to improve society by helping people without replacing them. To become an integrated part of our work, human workers must be able to teach cobots new tasks in a short time, making the robot a new tool in their toolbox. However, programming the cobot is most of the time done by experts and cobots cannot adapt to new task configurations, instead repeating learned patterns.

Learning from demonstration (LfD) (Rana et al., 2020)—a branch of learning focused on skill transfer and generalization through a set of demonstrations—enables cobots to learn and adapt motions from a set of demonstrations. State-of-the-art LfD methods require either manually tuning intrinsic parameters or a large amount of data, and have thus rarely been used in industrial contexts without experts, since manual tuning and data collection are time-consuming and error-prone. In this paper, we present TEAM (teach a robot arm to move), a novel method to learn from demonstrations *without manual tuning of intrinsic parameters* during training.

The main contributions of this paper are:

- A parameter-free framework to learn motions from a set of demonstrations, using a generative model to find a generalized trajectory, and attractor landscapes to reproduce the motion between different start and target joint angles.
- An optimization strategy of the attractor landscape's intrinsic parameters through Bayesian optimization.
- Improvement on the selection of the number of Gaussian Mixture Models through a series of one-tailed Welch's t-tests, based on the Jensen-Shannon divergence.
- Experimental validation of TEAM in two field tests showing that our method can be used by non-expert robot users.

A complete overview of the methodology is shown in Figure 1.

[a] https://orcid.org/0009-0004-9657-7249
[b] https://orcid.org/0000-0003-4430-3049
[c] https://orcid.org/0000-0003-1083-3180
[d] https://orcid.org/0000-0002-3079-0512

Panchetti, L., Zheng, J., Bouri, M. and Mielle, M.
TEAM: A Parameter-Free Algorithm to Teach Collaborative Robots Motions from User Demonstrations.
DOI: 10.5220/0012159700003543
In *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2023) - Volume 1*, pages 570-577
ISBN: 978-989-758-670-5; ISSN: 2184-2809

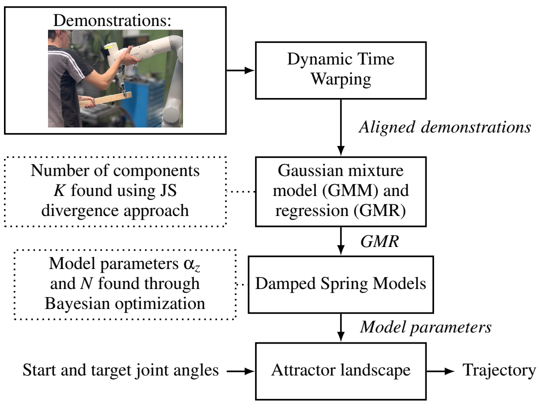

Demonstrations:
Dynamic Time Warping
Aligned demonstrations
Number of components K found using JS divergence approach
Gaussian mixture model (GMM) and regression (GMR)
GMR
Model parameters α_z and N found through Bayesian optimization
Damped Spring Models
Model parameters
Start and target joint angles
Attractor landscape
Trajectory

Figure 1: A set of demonstrations is recorded by a user and the motion is generalized through GMR. The system is modelled as a set of damped spring models that generalize the motion. Given start and target joint angles, model's parameters are used to generate new trajectories reproducing the motion taught by demonstrations. All system parameters are automatically optimized, and no expert knowledge is needed.

2 RELATED WORK

Rana et al. (2020) present a large-scale study benchmarking the performance of motion-based LfD approaches and show that Probabilistic Movement Primitives (ProMP) (Paraschos et al., 2013) methods are the most consistent on tasks with positional constraints. ProMP is a general probabilistic framework for learning movement primitives that allows new operations, including conditioning and adaptation to changed task variables.

Calinon et al. (2007) fit a mixture of Gaussians on a set of demonstrations and generalize the motion through Gaussian Mixture Regression (GMR) (Cohn et al., 1996). Trajectories are computed by optimizing an imitation performance metric. However, joint configurations are not constrained to the demonstration space, which can lead to the exploration of unsafe areas.

Kulak et al. (2021) propose to use Bayesian Gaussian mixture models to learn ProMP. While their method reduces the number of demonstrations needed to learn a representation with generalization capabilities, the method parameters must be manually set for all experiments.

Ijspeert et al. (2013) and Schaal (2006) use Dynamical Movement Primitives (DMP) to model complex motions through nonlinear dynamical systems. DMP is scale and temporal invariant, convergence is proven, but the parameters of the system must be manually tuned.

Pervez et al. (2018) propose a method that generalizes motion outside the demonstrated task space. Each demonstration is associated with a dynamical system, and learning is formulated as a density estimation problem. However, parameters must be set empirically for all dynamical systems.

Recent works have leveraged advances in deep learning. To tackle the challenging problem of model collapse, (Zhou et al., 2020) propose using a mixture density network (MDN) that takes task parameters as input and provides a Gaussian mixture model (GMM) of the MP parameters. During training, their work introduces an entropy cost to achieve a more balanced association of demonstrations to GMM components.

Pahič et al. (2020) propose to train a neural network to output the parameters of the DMP model from an image, before learning the associated forcing term. Pervez et al. (2017) use deep neural networks to learn the forcing terms of the DMP model for vision-based robot control. Both methods involve a convolutional neural network learning task-specific features from camera images. Sanni et al. (2022) estimate the correlation between visual information and ProMP weights for reach-to-palpate motion. The average error in task space is around 3 to 5 centimeters which is too high for our application. Yang et al. (2022) use reinforcement learning to learn a latent action space representing the skill embedding from demonstrated trajectories for each prior task. Tosatto et al. (2020) provide a complete framework for sample-efficient off-policy RL optimization of MP for robot learning of high-dimensional manipulation skills. All methods based on deep or reinforcement learning require a large amount of data. E.g., Sanni et al. (2022) show the robot the reach-to-palpate motion 500 times, Pervez et al. (2017) acquire 50 demonstrations for a single task, and Yang et al. (2022) uses around 80K trajectories.

3 METHOD

3.1 Overview

To learn a motion, a set of demonstrations is first collected by the user. In our work, the cobot is taught by manual guidance—see the image in the demonstration box of Figure 1. A demonstration stores the cobot's joint angles recorded while the cobot is shown the task and the robot is controlled in joint space to avoid singularities during the motion of the redundant robot arm. As in previous work by Calinon et al. (2007), demonstrations are aligned in time using dynamic time warping (Sakoe and Chiba, 1978).

The first step of our method consists in finding the best Gaussian mixture model (GMM) fit on the demonstrations dataset and calculates the Gaussian Mixture Regression (GMR)—i.e. the generalized trajectory. Section 3.2 shows how to use the Jensen-Shannon (JS) divergence (Lin, 1991) to fit the GMM and GMR without user input. From the GMR, the motion is represented as a set of damped spring models; Section 3.4 shows how to estimate the optimal parameters of the models through Bayes optimization. Finally, the optimal motion is computed by the attractor landscape, given initial and goal cobot joint angles.

3.2 Gaussian Mixture Model and Gaussian Mixture Regression

Given a set of demonstrations, a GMM is fitted on each degree of freedom—each of the cobot's joints. Maximum likelihood estimation of the mixture parameters is done using Expectation Maximization (EM) (Dempster et al., 1977).

The number of mixture model components k is critical to obtaining a GMM leading to a smooth GMR. While Calinon et al. (2007) used the BIC criterion to determine the optimal value k—denoted k^* in our work—Pervez et al. (2017) showed that BIC overfits the dataset without a manually tuned regularization factor. We propose a novel strategy to find k^* without any manual thresholds, based on cross validation, the JS divergence, and statistical analysis.

For $k = 2$ until $k = c$—with c the maximum number of components in the GMM—50 cross validations are performed over the demonstration dataset using the JS divergence as a measure of similarity between the GMMs generated from the train and test splits. The mean m_k and standard deviation s_k of the JS divergences for each k are stored in the set $s(k) \rightarrow (m_k, s_k)$. k^* is initialized as the value in s with the minimum m_k. For each key $k \in s$, a serie of one-tailed Welch's t-tests (Welch, 1947) with $\alpha = 0.05$—i.e. there is a 5% chance that the results occurred at random—is used to evaluate whether k is a more optimal number of components than the current value of k^*. First, we test if the JS divergence of k is strictly greater than that of k^* . The null hypothesis H1 and alternative hypothesis H2 are:

Hypothesis 1 (H1): $k - k^* \leq 0$

Hypothesis 2 (H2): $k^* - k < 0$

If the null hypothesis is rejected, k is strictly greater than k^* and k is *not* the optimal number of components. If we fail to reject the null hypothesis, we then test if k is strictly less than k^*. The null hypothesis H3 and alternative hypothesis H4 are:

```
   Data: demonstrations set $D$
   Result: $k^*$
 1 $s \leftarrow$ empty map;
 2 for $k = 2$ until $k = c$ do
 3     $res \leftarrow$ empty list;
 4     for 1 to 50 do
 5         Sample datapoints of $D$ in two equal
            sets $D_1$ and $D_2$;
 6         $G_1 \leftarrow$ GMM with $k$ components fitted
            on $D_1$;
 7         $G_2 \leftarrow$ GMM with $k$ components fitted
            on $D_2$;
 8         Add $JSdivergence(G_1, G_2)$ to $res$;
 9     end
10     $m_k, s_k \leftarrow$ mean(res), std(res);
11     $s(k) \leftarrow (m_k, s_k)$;
12 end
13 $k^* \leftarrow$ component in $s$ with the lowest mean;
14 for key $k$, value $(m_k, s_k) \in s$ do
15     if H1 is not rejected then
16         if H3 is rejected or $s_k < s_{k^*}$ then
17             $k^* \leftarrow k$;
18         end
19     end
20 end
21 return $k^*$
```

Algorithm 1: Algorithm used to determine the best number of components for the GMM.

Hypothesis 3 (H3): $k^* - k \leq 0$

Hypothesis 4 (H4): $k - k^* < 0$

If the null hypothesis is rejected, k^* is strictly greater than k, and k is the optimal number of components. If we failed to reject both H1 and H3, no conclusions as to whether k or k^* is the best estimate can be drawn, and k^* is set to the most stable number of components: $k^* = k$ *if and only if* s_k is lower than s_{k^*}.

The process to determine the optimal number of Gaussians is detailed in Algorithm 1.

3.3 Damped Spring Model

TEAM uses the damped spring model formulated by Ijspeert et al. (2013): :

$$\begin{aligned} \tau\dot{z} &= \alpha_z(\beta_z(g-y)-z)+f \\ \dot{y} &= z \end{aligned} \tag{1}$$

where τ is a time constant, f is the nonlinear forcing term, α_z and β_z are positive constants, and g is the target joint angles. The forcing term f of Equation (1) is used to produce a specific trajectory—i.e. the GMR. Since f is a nonlinear function, it can be represented as a normalized linear combination of basis functions (Bishop, 2006):

$$f(x) = \frac{\sum_{i=1}^{N} \Psi_i(x)\omega_i}{\sum_{i=1}^{N} \Psi_i(x)}(g - y_0)v \tag{2}$$

where Ψ_i are fixed radial basis functions, ω_i are the weights learned during the fit, g is the goal joint angles, and v is the system velocity. N is the number of fixed radial basis function kernels $\Psi_i(x)$. Detailed derivations, and methods to compute ω_i and the joint dynamics, are found in Ijspeert et al. (2013).

3.4 Parameters Optimization

For y to monotonically converge towards the target g, the system must be critically damped on the GMR by choosing the appropriate values of α_z and β_z. As shown by Ijspeert et al. (2013), β_z can be expressed with respect to α_z as $4\beta_z = \alpha_z$. Thus, only two parameters control the tracking of the reference and the stability: the number of radial basis functions N and the constant α_z. In the previous state-of-the-art (e.g. Ijspeert et al. (2013) and Pervez et al. (2018)), α_z and N are empirically chosen by the user. Instead, TEAM uses Bayesian optimization (BO) (Garnett, 2022) to determine α_z and N and avoid manual tuning.

The error to minimize is the sum of both the root mean squared error with respect to the GMR and the distance of the trajectory endpoint with respect to the goal reference:

$$f(\alpha_z, N) = \sqrt{\frac{\sum_{t=1}^{T}(y(\alpha_z, N, t) - y_G(t))^2}{T}} + ||y(\alpha_z, N, T) - y_G(T)|| \tag{3}$$

where $y(\alpha_z, N, t)$ is the joint angles at time t obtained with DMP parameters α_z and N, $y_G(t)$ is the GMR joint angles at time t, and $||\cdot||$ is the l_2-norm. The acquisition function is the expected improvement (EI):

$$EI_i(x) := E_i[f(x) - f_i^*] \tag{4}$$

where $E_i[\cdot|x_{1:i}]$ indicates the expectation taken under the posterior distribution given evaluations of $f(x)$ at $x = x_1, \ldots, x_i$. The acquisition function retrieves the point in the search space that corresponds to the largest expected improvement and uses it for the next evaluation of the objective function $f(x)$. The point x_i minimizing the value of $f(x)$ corresponds to the optimal combination of α_z and N. The optimization is stopped when two successive query points are equal.

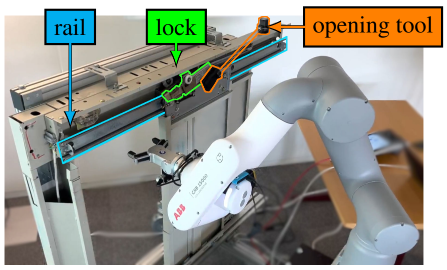


Figure 2: The cobot faces the test elevator door used for evaluation of TEAM in a maintenance scenario.

4 EVALUATION

We evaluate our method in two real-world scenarios, using a 6-axis ABB GoFa CRB 15000 cobot[1]—pose repeatability at the maximum reach and load is 0.05 mm. In the first scenario, the cobot works alone to do maintenance operations on an elevator door. This scenario is used to evaluate the stability of the method—both the parameter selection and its robustness to start and goal angle changes. The second scenario pertains to the ease of use of our method for non-expert users: three Schindler workers teach the cobot a set of tasks needed to drill elevator pieces on Schindler's factory line.

The desired workflow for field technicians is one where, for a given task, the cobot first learns the motion and then reproduces the motion on the factory line without having to be trained again. Hence, for each task, a set of demonstrations is recorded by a user and the cobot learns the motion using TEAM. Then, using the previously trained model, the cobot reproduces the task multiple times with different start joint angles.

To measure the cobot's accuracy in reaching the target joint angles, we measure the mean absolute error e_j between the goal joint g and actual end joint t angles:

$$e_j = \frac{\sum_{i=1}^{n} |t_i - g_i|}{n} \tag{5}$$

with n the number of joints. To measure the quality of the reproduced motion, we use the Generalized Multiple Correlation Coefficient (GMCC) proposed by Urain et al. (2019), a measure of similarities between trajectories that is invariant to linear transformations. Code, datasets, and metrics can be found online. [2]

4.1 Door Maintenance Dataset

The 5 tasks of elevator door maintenance dataset:

[1]https://new.abb.com/products/robotics/collaborative-robots/crb-15000

[2]https://github.com/SchindlerReGIS/team

Table 1: JS divergence repeatability over 50 runs.

Task	T1	T2	T3	T4	T5
Median nb GMM	5 ± 0.40	3 ± 0	4 ± 0	3 ± 0	4 ± 0.27

Table 2: Comparison between grid search (GS) and Bayesian optimization (BO). Statistics over 50 runs.

Task	T1	T2	T3	T4	T5
GS minimum	12.84	73.91	30.88	36.58	26.44
BO minimum	12.84 ±0	74.22 ±0.61	30.98 ±0.40	36.68 ±0.27	26.44 ±0.05
GS time [s]	2348.97	2600.96	2427.34	2149.50	1848.20
BO median time [s]	19.39 ±3.63	27.40 ±10.17	12.40 ±2.40	26.11 ±18.54	12.53 ±1.99
GS calls	3750	3750	3750	3750	3750
BO calls	23.14 ±3.88	34.20 ±10.85	16.96 ±2.73	32.98 ±18.97	20.14 ±2.58

- $T1$: open and lock the door using a custom opening tool. The lock is now in the middle of the rail.
- $T2$: grab the cleaning tool.
- $T3$: clean the rail while avoiding the lock. The cobot must aim for both ends of the rail with the brush since most dust accumulates there.
- $T4$: drop the cleaning tool on its support.
- $T5$: grab the opening tool, close the door, and combine the two pieces of the opening tool.

The maintenance setup can be seen in Figure 2.

4.1.1 Evaluation of the Parameters' Repeatability

A repeatability analysis of the parameters K, α_z, and N, is done on the data collected for the door maintenance scenario.

Repeatability of K Using the JS Divergence: the method described in Section 3.2 is run 50 times for each task in the maintenance dataset—with a minimum of 2 Gaussians and a maximum of 9. As seen in Table 1, the median number of Gaussians for each dataset varies only by a small standard deviation, showing that the selection of the number of Gaussian components is stable.

Damped Spring Model Parameters: we compare BO with grid search (GS) for 50 runs per task in the maintenance dataset. One can see in Table 2 that BO converges 100 times faster than GS and to the same global optimum. Optimization took an average of 19.57s on an Intel Core i5 10th Gen and there is a reduction by a factor of at least 100 in the number of iterations needed—it should be noted that larger standard deviations in the running time are usually due to larger outliers with a median time around 20s.

In conclusion, we find that the JS divergence and BO lead to a stable selection of K, α_z, and N, and can be used as sensible replacements for the manual tuning previously done by expert users.

4.1.2 Adaptability

Table 3 shows the number of demonstrations recorded for each task, with the average training times and error metrics.

The complexity of DTW is $O((M-1)L^2)$, with M the number of demonstrations in the dataset and L the longest demonstration length, GMM and GMR are $O(KBD^3)$ where B is the number of datapoints in the dataset, D the data dimensionality, and K the number of GMM components. The Gaussian Process of the BO is $O(R^3)$ with R being the number of function evaluations—Table 2 shows that R is at worse around 34.20 ± 10.85. As seen in Table 3, in the maintenance scenario, the maximum training time is under 4min—the longest training time is 203.33 ± 4.32s for $T2$.

Once a model is trained, the accuracy of the reproduced trajectory is evaluated by computing 30 reproductions and calculating the GMCC between the reproduced trajectory and the GMR. For each reproduction, the target joint angles are the same as the last joint angles of the GMR, and the start joint angles are the same as the GMR, with the addition of a zero mean Gaussian noise with standard deviation of 1, 5, 10, and 20 degrees. For each noise value, the average GMCC and e_j per task are shown in Table 3. One can see that, regardless of the noise value, GMCCs and e_j are very close to 1 and 0 respectively, showing that the cobot accurately reproduces the demonstrated motion and reaches the target joint angles. E.g., Figure 3 shows the regression trajectory and reproduction per joint with Gaussian noise with standard deviation of 20 deg: the trajectory of each joint is conserved regardless of the noise added to the start joint angles.

4.2 Field Tests and User Study

To validate that the cobot can be used by non-expert users in a professional setting, we conducted field tests at the Schindler headquarters with three Schindler field workers working on the production line. None of the workers had worked with a cobot before. To ensure realism of the tasks, the field workers designed four test scenarios that would reduce their workload *if* the robot can easily be taught how to perform the task:

- $F1$: find a metal piece, grab and place it on a drilling machine.
- $F2$: find a metal piece, grab and place it on a drilling machine while avoiding an obstacle.

Table 3: This table presents the training time and error metrics for the maintenance tasks— 30 runs per task.

Noise	Task	T1	T2	T3	T4	T5
	Number of demonstrations	6	4	4	4	3
	Average demonstration duration [s]	33.19 ± 2.38	36.37 ± 2.28	32.93 ± 3.00	28.17 ± 3.33	26.19 ± 1.05
	Training time [s]	184.08 ± 2.45	203.33 ± 4.32	185.82 ± 3.07	162.89 ± 4.56	148.59 ± 4.51
1 deg	GMCC	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	e_j [deg]	0.14 ± 0.00	0.19 ± 0.00	0.30 ± 0.00	0.01 ± 0.00	0.05 ± 0.00
5 deg	GMCC	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	e_j [deg]	0.14 ± 0.00	0.19 ± 0.00	0.30 ± 0.00	0.01 ± 0.00	0.05 ± 0.00
10 deg	GMCC	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	e_j [deg]	0.14 ± 0.00	0.19 ± 0.00	0.28 ± 0.00	0.01 ± 0.00	0.05 ± 0.00
20 deg	GMCC	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	e_j [deg]	0.14 ± 0.00	0.19 ± 0.00	0.28 ± 0.00	0.015 ± 0.00	0.05 ± 0.00

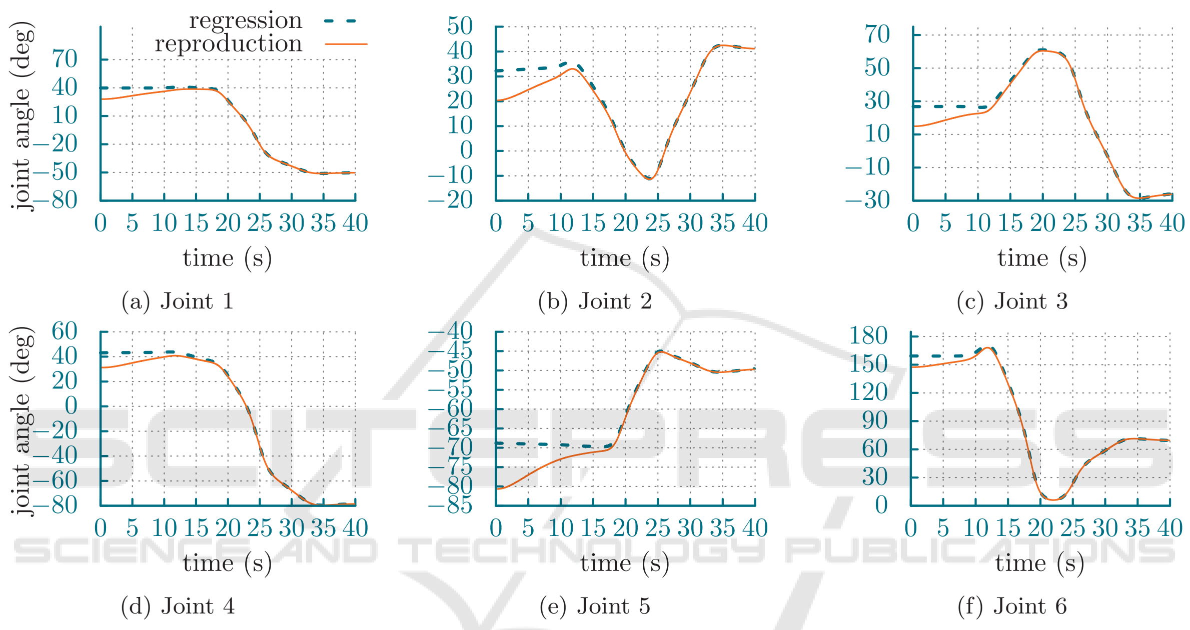


Figure 3: Reproduction and regression joint angles evolution on an example of the $T1$ task. Start joint angles of the reproduction are computed by adding Gaussian noise with 20deg standard deviation on the regression initial joint angles. The reproduced trajectory reproduces the regression's motion and reach the target joint angles.

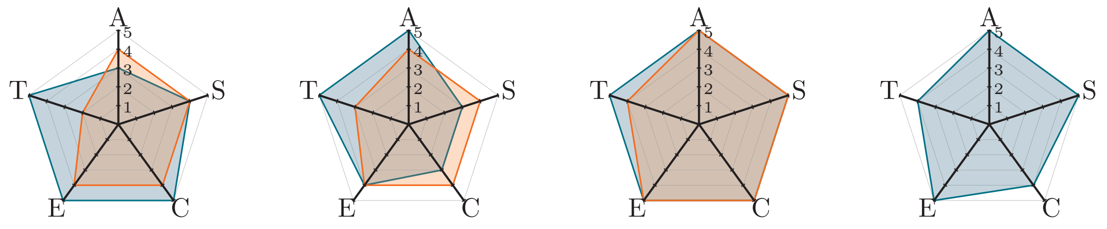
Table 4: This table presents error metrics for the factory scenario—dataset consisted of 3 to 4 demonstrations.

Task	F1	F2	F3	F4
Number of reproductions	31	30	33	20
GMCC	0.99 ± 0.02	0.99 ± 0.00	0.99 ± 0.01	1.00 ± 0.00
Joints error e_j [deg]	0.00 ± 0.00	0.02 ± 0.02	0.04 ± 0.05	0.02 ± 0.02

- $F3$: find a metal frame, grab and place it on a drilling machine while rotating the piece.
- $F4$: find a wooden plank, grab one side while the worker grabs the other, and place it together on a drilling machine.

A custom app on a smartphone was used by the workers to interact with the robot in an intuitive manner. The app consists of two main pages: one to record demonstrations and train a model, and another page to give the robot a target position and start the task reproduction. After a short training on how to use the app, three to five demonstrations were recorded per user, per task. To calculate the metric, each task was reproduced around 10 times per user, apart from $F4$ where only two users participated—hence 20 reproductions. Detection of the different objects is done using template matching (Brunelli, 2009).

Table 4 shows the GMCC and e_j for all tasks, calculated for 30 reproductions of the motion for each task. The error metrics results are similar to the ones presented in Section 4.1, with GMCC averaging 0.99 and e_j of 0 deg; demonstrating accurate task reproduction in a realistic scenario.



(a) User 1, day 1 (b) User 1, day 2 (c) User 2 (d) User 3

Figure 4: After use, the cobot is evaluated by each of the users on: safety (S), easiness to teach (T), entertainment (E), reaching the target joint angles (A), and task completion (C). In blue the results for the task F1, F2, and F3, and in red the results for the collaborative task F4. One can see the collaborative task, where the user carries a piece of wood with the robot, is more difficult than other task were the cobot works next to the user.

The field tests were conducted over two days and, at the end of each day, the workers answered a questionnaire to evaluate the cobot's performance. In the survey, users rate the following statements on a scale from 1 to 5, corresponding to strongly disagree, disagree, neutral, agree, and strongly agree:

- The cobot learned the correct motion.
- I felt safe operating the cobot.
- The cobot reached the goal point accurately.
- Teaching the cobot a motion was simple.
- Teaching the cobot a motion was entertaining.

The radar plots in Figure 4 present survey results. While users showed satisfaction with the cobot's precision and motion performance, the complexity of holding the beam and moving the cobot while showing the motion in F4 led to a lower score for easiness of teaching compared to other tasks.

5 LIMITATIONS AND FUTURE WORK

TEAM doesn't consider elements of the environment during the motion. This create confusion for the workers not understanding why the cobot does not avoid obstacles, making it harder for them to trust the cobot. Future work will look at integrating visual information through cameras to update the motion depending on the environment.

Another way that TEAM could be improved is by being able to update the attractor landscape of a motion incrementally. Future work will look into making the process incremental, giving workers the ability to correct existing motions learned by the cobot.

6 SUMMARY

A method to learn motions from demonstrations requiring no manual parameter tuning has been developed. Given a set of demonstrations aligned in time, the motion is generalized using GMM and the reference trajectory is extracted with GMR. Since BIC criterion can lead to over-fitting of the GMM, it is proposed to instead use the Jensen-Shannon divergence to determine the optimal number of GMM components. The cobot DOFs are represented as damped spring models and the forcing term is learned to adapt the motion to different start and goal joint poses. Parameters of the spring model are found using Bayesian optimization.

TEAM is extensively evaluated in two field tests where the cobot performs tasks related to elevator door maintenance, and works in realistic scenarios with Schindler field workers. The precision in joint angles and motion reproduction quality are evaluated, and the experiments show that the cobot accurately reproduces the motions—GMCC and mean average error for the final joint angles are around 1 and 0 respectively. Furthermore, feedback collected from the field workers shows that the cobot is positively accepted since it is easy to teach and easy to use.

REFERENCES

Bishop, C. M. (2006). Pattern recognition and machine learning (information science and statistics).

Brunelli, R. (2009). Template matching techniques in computer vision: Theory and practice.

Calinon, S., Guenter, F., and Billard, A. (2007). On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37:286–298.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. In *NIPS*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Garnett, R. (2022). *Bayesian Optimization*. Cambridge University Press. in preparation.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. (2013). Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation*, 25(2):328–373.

Kulak, T., Girgin, H., Odobez, J.-M., and Calinon, S. (2021). Active learning of bayesian probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 6:2163–2170.

Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

Pahič, R., Ridge, B., Gams, A., Morimoto, J., and Ude, A. (2020). Training of deep neural networks for the generation of dynamic movement primitives. *Neural networks : the official journal of the International Neural Network Society*, 127:121–131.

Paraschos, A., Daniel, C., Peters, J., and Neumann, G. (2013). Probabilistic movement primitives. In *NIPS*.

Pervez, A. and Lee, D. (2018). Learning task-parameterized dynamic movement primitives using mixture of gmms. *Intelligent Service Robotics*, 11:61–78.

Pervez, A., Mao, Y., and Lee, D. (2017). Learning deep movement primitives using convolutional neural networks. *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 191–197.

Rana, M. A., Chen, D., Ahmadzadeh, S. R., Williams, J., Chu, V., and Chernova, S. (2020). Benchmark for skill learning from demonstration: Impact of user experience, task complexity, and start configuration on performance. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7561–7567.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26:159–165.

Sanni, O., Bonvicini, G., Khan, M. A., López-Custodio, P. C., Nazari, K., and AmirM.Ghalamzan, E. (2022). Deep movement primitives: toward breast cancer examination robot. *ArXiv*, abs/2202.09265.

Schaal, S. (2006). *Dynamic Movement Primitives -A Framework for Motor Control in Humans and Humanoid Robotics*, pages 261–280. Springer Tokyo, Tokyo.

Tosatto, S., Chalvatzaki, G., and Peters, J. (2020). Contextual latent-movements off-policy optimization for robotic manipulation skills. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10815–10821.

Urain, J. and Peters, J. (2019). Generalized multiple correlation coefficient as a similarity measurement between trajectories. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1363–1369.

Welch, B. L. (1947). The generalization of ‘student’s’ problem when several different population varlances are involved. *Biometrika*, 34:28–35.

Yang, Q., A. Stork, J., and Stoyanov, T. (2022). Mpr-rl: Multi-prior regularized reinforcement learning for knowledge transfer. *IEEE Robotics and Automation Letters*, pages 1–8.

Zhou, Y., Gao, J., and Asfour, T. (2020). Movement primitive learning and generalization: Using mixture density networks. *IEEE Robotics & Automation Magazine*, 27:22–32.