Robot Imitation Learning from Kinesthetic Teaching

Jack Landers

I. Introduction

In this report, we explore Dynamic Movement Primitives, Multiscale Dynamic Time Warping, and Gaussian Mixture Regression as techniques for robot learning. We collected data for cartesian positions mapping trajectories of a manipulator as a complex task was performed by multiple participants. More demonstrations require more data collection, deeper data analysis, and further computations, but can be useful for creating a robust model that can generalize. The 7-DOF robotic arm was controlled physically while in gravity compensation mode, so that we could manually demonstrate the desired trajectories for performing the task. Our goal is to utilize this data to develop a generalized control policy, whereby the robot can perform this task for any initial and end position, and avoid any obstacles detected in its path.

II. Vizualizing the Data

A. Displaying a Demonstration

Altogether we collected eleven demonstrations of the trajectories each individually represented in a csv file with timestamps for readings of cartesian positions (x, y, and z,) and quaternion rotation (qw, qx, qy, qz). We begin by plotting a singular demonstration to understand this data.

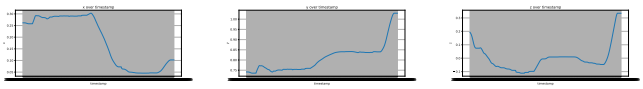
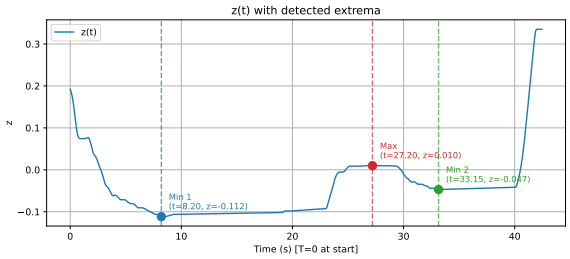


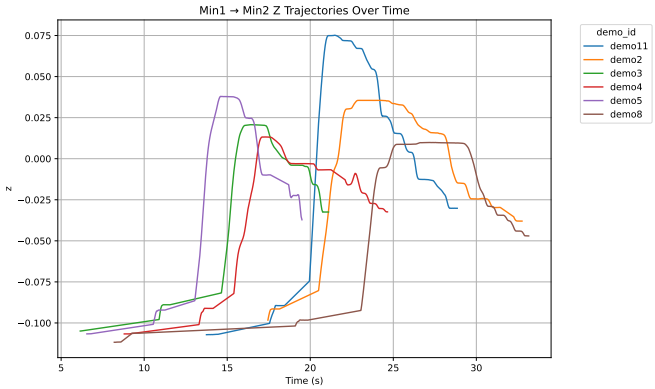
Fig. 1: x, y, z for a single pick and place demonstration

B. Finding Points of Interest

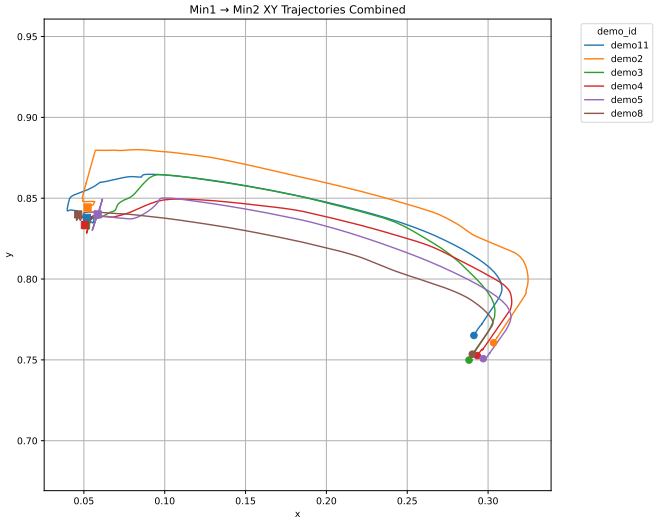
While quaternion data is especially hard to interpret, the z data is most recognizable as our task was to pick and place. For this reason, we define the moments of action when we pick and place as 2 local minima in the z data. To extrapolate these points of interest we convert the timestamps to timesteps and use a threshold model to identify significant extrema with the pick action at z-minimum in the first 20 seconds of demonstration, the maximum between 10 seconds and 30 seconds, and the place as our final z-minimum after that. Defining these extrema, we can then pre-process our data for learning, by removing noisy demonstrations and using the picking and placing z-values as initial and end positions, respectively. This process filtering the data to procure the trajectories from our complete demonstration data is shown in Figure 2.



(a) Finding Extrema in the Altitude



(b) Z-axes of the Pick and Place Demonstrations



(c) Overhead of the Pick and Place Demonstrations

Fig. 2: Preprocessing the Trajectories Data

Professor Maria Kyrarini, Santa Clara University
Submission deadline February 2, 2026

III. Dynamic Movement Primitives

Dynamic Movement Primitives (DMP) describes converging a mechanical system to a given target to perform complex behaviors in real-time. They use an added forcing term to a stable behavior in order to produce a complex trajectory. The learnable autonomous forcing term transforms simple dynamical systems into weakly nonlinear ones such that the trajectory would always converge towards that demonstrated and is zero initially. Once we introduce a new initial or goal state, the path is discrete from point to point and modified by an attractor. DMPs are especially effective at adaptively repeating simple actions in novel contexts like target switching, by reducing over-amplifications in the trajectory. They are also reliable due to their nature as bounded input bounded output stable systems that are invariant to rescaling.

$$f(s) = \tau^2 \ddot{x} - \alpha\left(\beta(g - x) - \tau \dot{x}\right) \tag{1}$$

Where:

- $x, \dot{x}, \ddot{x}$ are the position, velocity, and acceleration,
- τ is the temporal scaling factor,
- α, β are the spring damper gains, and
- $f(s)$ is the nonlinear force in terms of phase.

A. PyDMP

To visualize the Dynamic Movement Primitives we used the PyDMP library by AlexanderFabisch. With this we are treating our trajectory as a spring damper system by calculating the nonlinear forcing term that defines the shape of the movement over time from phase using Equation 1. Figure 3 shows that the DMP learns to imitate a singular trajectory movement. Such learning attractor models are vital for repetitive robotic tasks in dynamic environments that may include modifications in the boundaries, constraints, or obstacles posed to the robot.

B. Testing Generalization

To test the adaptiveness of our DMP model, we introduce offset to our initial and end position, to determine if the output trajectory still follows a similar shape. In Figure 4 it is evident that the model has generalized the demonstration as the trajectories both scale and translate.

IV. Dynamic Time Warping

Before we can align multiple trajectories, it must be considered that our pick and place demonstrations were not performed at identical speeds, and that the same movements can take place at different paces across the trajectory. In order to synchronize the data in an optimal way we use Dynamic Time Warping (DTW) which evaluates corresponding differences in both time and space. This requires elastic alignment, matching shapes even without phase, where multiple positions in x may correspond to a single position in y or vice-versa. Our optimal path here must retain path bounds both monotonicity and sequentialism to represent the minimum distance between previous points, and the Euclidian distance of the next point.

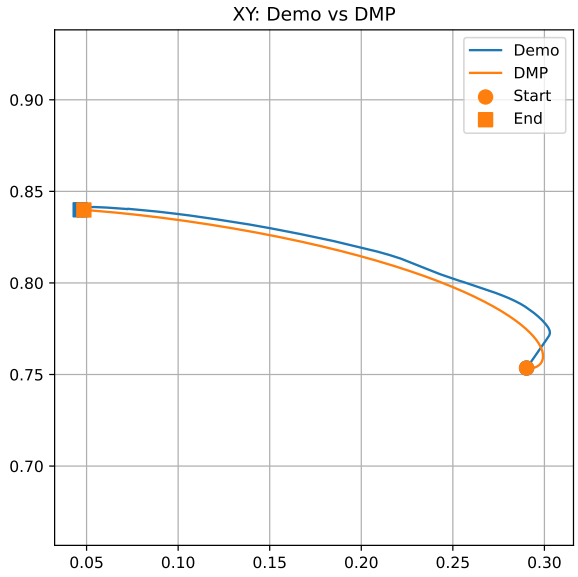


Fig. 3: Applying the DMP

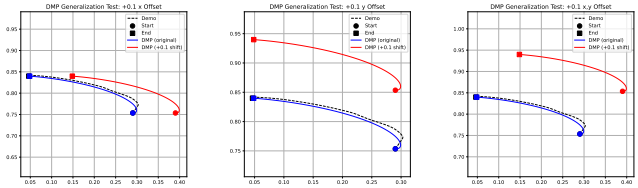


Fig. 4: Testing Generalization for Offsets in x, y, and Both Axes.

A. Distance Matrix

To temporally align different demonstrations, we calculate the Euclidean distance cost of their trajectories for discrete timesteps and display them in a matrix. In Figure 5 we visualize the most closely aligned timelines, and can exclude demos 2 and 11 as outliers.

B. Applying DTW

We start with the nearest demonstrations and apply the DTW using the dtw-python library to produce a warping path showing the timing difference between demos 3 and 4 in Figure 6.

C. Multiscale DTW

For comparison, we also developed a Dynamic Time Warping strategy that utilizes multiscaling as outlined in Chapter 4 of 'Information Retrieval for Music and Motion' by Springer, Berlin, and Heidelberg. The technique accelerates the calculation by combining DTWs of different resolutions to constrain the path at high resolutions, where the computation increases exponentially. Figures 7 and 8 show the process of resampling the trajectory and producing a weighted representation of the

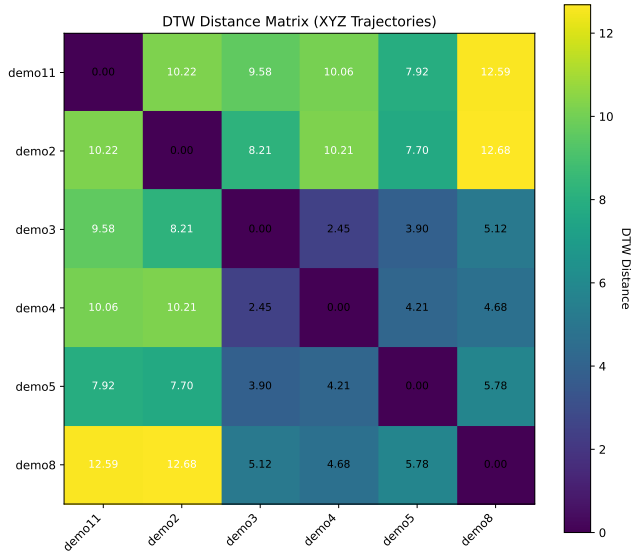


Fig. 5: Distance Matrix

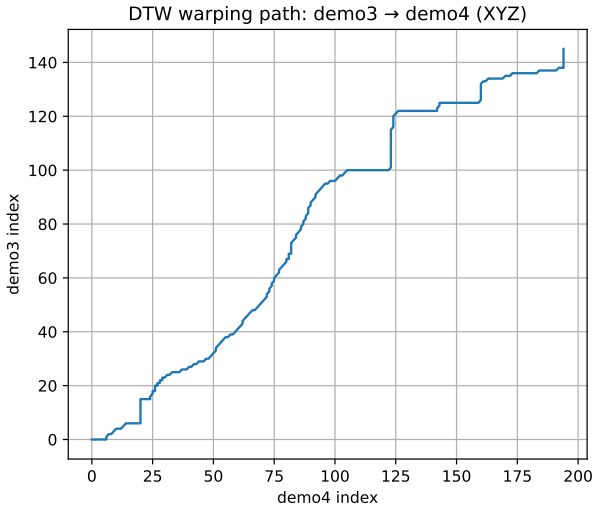


Fig. 6: Warping Path

warping path, with the band of local calculated distances highlighted. This process is iterated from lower samples to higher resolutions, continuing to bound the path. Fewer points have to be unnecessarily computed, at the cost of representing sharp gradients in the speed of action, more smoothly. We see the smoothing of these gradients when comparing the warping paths shown in Figure 9 and Figure 6. By computing the DTW between each trajectory and collecting them, we indicate temporally aligning samples as points along the trajectories in Figure 10. Multiscaling is especially effective for our case where we do not want rapid motions to be represented in our trajectory, but should be noted as a limitation across applications.

By resampling within thresholded pixels, we rebuild the warping path up to full resolution. There, we can see the effect of resampling the DTW in that the steeper gradients of our full

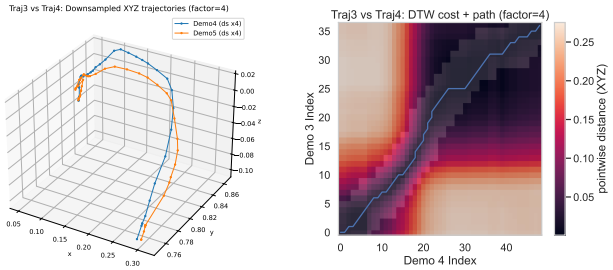


Fig. 7: DTW of demos downsampled by a factor of 4

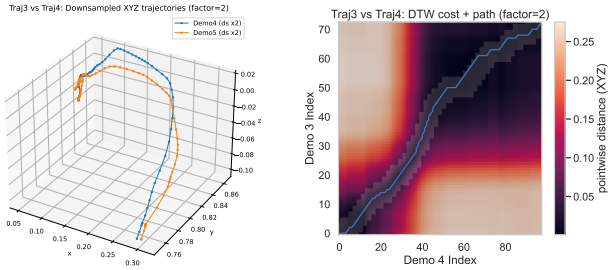


Fig. 8: DTW resampled up to a factor of 2 within a local band

warping path are now smoothed, but the number of operations required is significantly reduced.

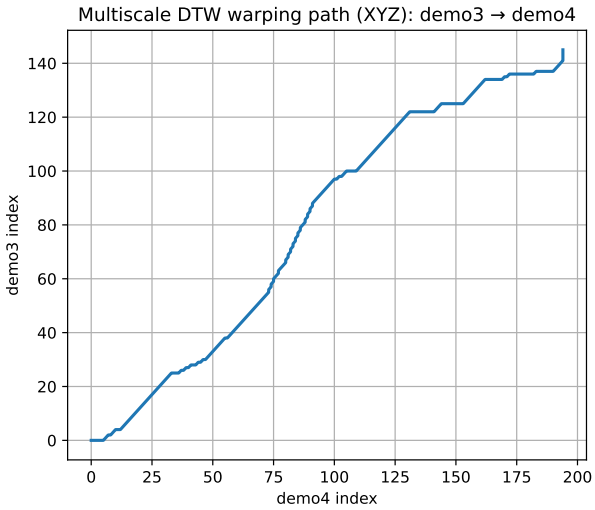


Fig. 9: Multiscale Warping Path

We apply this multiscale DTW to all of the remaining trajectories, aligning them.

V. Gaussian Mixture Regression

To optimally perform behavior cloning from multiple kinesthetic teaching demonstrations, we represent the trajectories with a gaussian mixture model. A combination of gaussian distributions describes the mean and variance of the data throughout the motion. The mean follows the general pathway of the task action, while the variance identifies where the trajectories are restricted to align at certain points such as

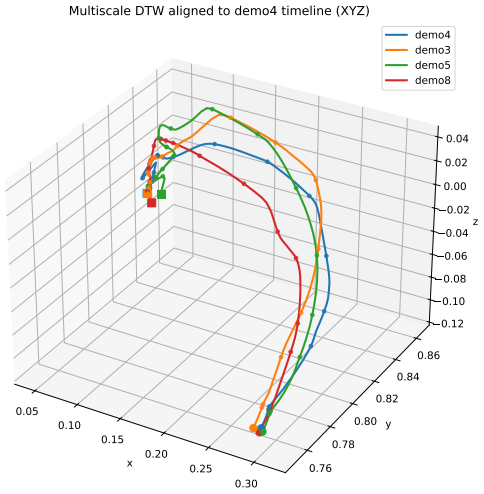


Fig. 10: Multiscale DTW Applied to Multiple Trajectories

the pick and place points of interest in our demonstrations. Together, this reduces the amount of data required to represent the trajectories so that the parameters can be estimated via expectation maximization. We define a loss function and adjust the parameters to converge towards the optimal control policy.

A. Bayesian Information Criterion

Before computing the probabalistic model of our data we first need to determine the model complexity so that we can accurately represent it without overfitting and memorizing noise. Bayesian Information Criterion (BIC) defines the number of parameters required by balancing the likelihood that they will fit N data points with a penalty term that discourages unnecessary gaussians. The minimum BIC term is found by iterating for an increasing number of gaussians until it is no longer growing as shown in Figure 11. We calculate BIC with the sklearn library where we can pass a gaussian mixture model for it to compute for us.

$$BIC = -2LogL(\hat{\theta}) + kLogN \tag{2}$$

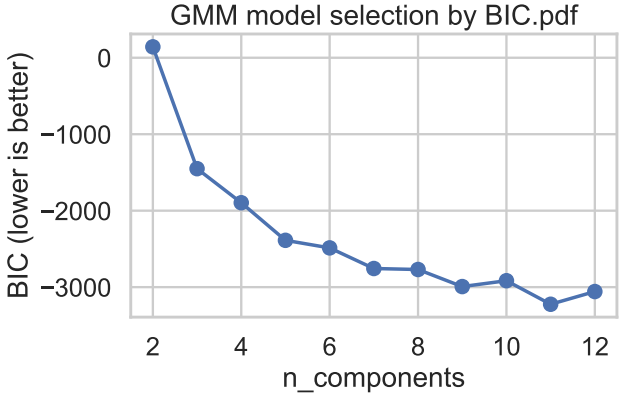
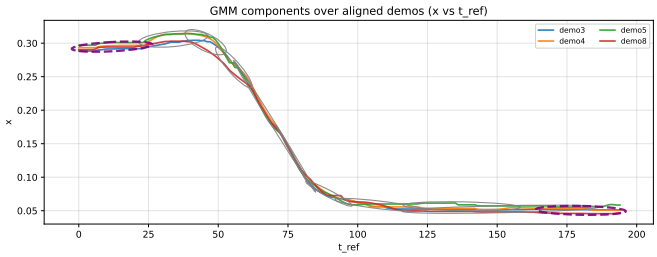


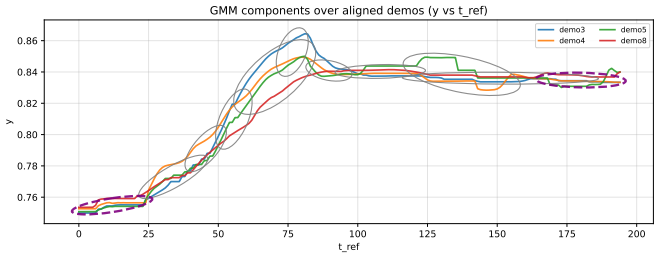
Fig. 11: BIC

B. Gaussian Mixture Model

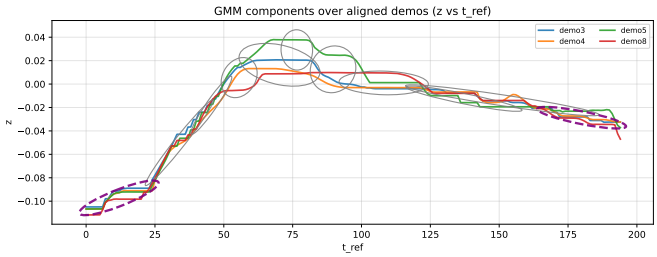
With the optimal number of components computed by the BIC we can then fit the Gaussian Mixture Model (GMM) to the set of trajectories. In this case we see 11 gaussians fit across all three dimensions. Doing so you produce a vector of means per gaussian, but also a covariance matrix defining not only the variance of the data across trajectories but also across dimensions to show correlations in both shape and orientations. For Figure 12 we display the gaussians and highlight those most dense at the mean where the covariance matrix is minimized. Such points correlate with our pick and place extrema where the movements become more precise and calculated to specifically perform the endpoint grasp and release task rather than just transferring the object.



(a) x vs t_{ref}



(b) y vs t_{ref}



(c) z vs t_{ref}

Fig. 12: GMM components over aligned demonstrations

C. Gaussian Mixture Regression

The GMM is a fitted statistical representation of the complex dataset of values that constructs the demonstrations. From this, we can perform regression, the supervised machine learning process of fitting a model to meet target values. We apply optimization to adjust weights and minimize the cost function which quantifies the difference from the goal. From the input, the influence of each gaussian on the trajectory policy is learned to meet the output with the covariance describing

confidence. Fusing these output gaussian distributions gives a predicted trajectory following our learned behavior.

Our GMR produces a stable mean along the path from the initial to the end goal position. Figure 13 shows a GMR control policy for the equivalent positions overlayed on the demonstrations that is naturally smoother than any of the human guided demonstrations. Furthermore, Figure 16a, b, and c depict how the model generalizes to adjusting the initial and endpoints and following a simiar trajectory still.

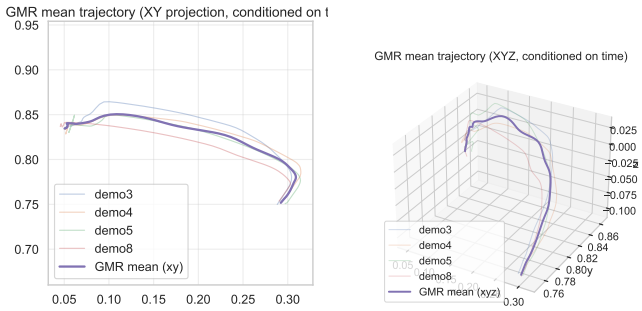


Fig. 13: GMR Path

D. Piecewise GMM

For advanced tasks, GMR can fail to learn, so we use high-level information on the task to split the data into multiple trajectories for more accurate GMM. In the case of our pick and place demonstrations, we only have a single operation, and still it was vital we used our understanding of the data to evaluate where the start and finish positions were occurring and reduce any additional noise. If we were to also perform another action in the same demonstration, for the control policy to effectively learn, we would have to break down the tasks into the high level operations so that GMM is only fit to one task at a time.

VI. Obstacle Avoidance with GMR

To further improve our imitation learning policy, we introduce a control adaptation for avoiding obstacles in a circular manner, by describing a repulsive field sourced from its centerpoint. The force vector is a function of the inverse distance from the obstacle when within a threshold range, adjusting the trajectory to move away while maintaining any momentum from the path in other directions.

A. Example of Circular Avoidance

We first tested our policy in 2-dimensions with a simple sine wave representing the GMR trajectory. From this, we find that the model can successfully avoid the object, but to do so in an acceptable manner, we need to introduce smoothing so that the robot is not taken off its predicted trajectory so sharply. Immediate deviations in the target direction can produce large forces when the rigid body has a high mass or under high capacity load, which increases demand on hardware and introduces further safety constraints.

B. Avoidance in 3D

To effectively perform the smoothing of the avoidance in 3-dimensions, we integrate a Laplacian smoothing filter which uses second order smoothing. A second order filter counters the high gradient distortions in acceleration and develops the desired smooth curvature around the obstacle that we see applied to our computed GMR in Figures 14 and 15.

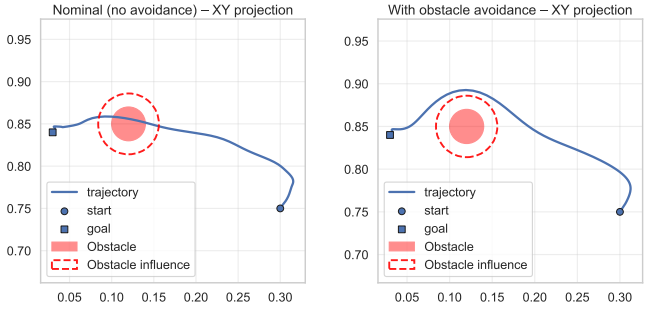


Fig. 14: Applying Circular Avoidance in xy

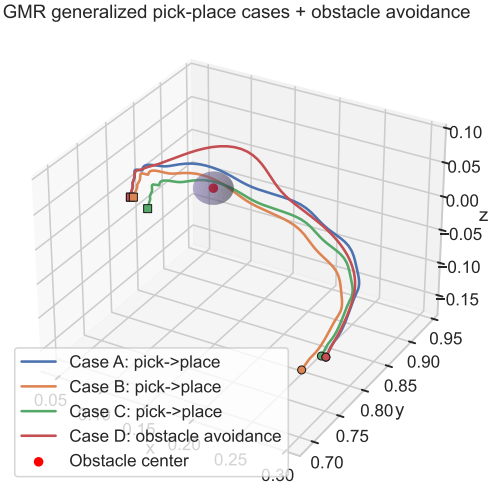


Fig. 15: GMR Generalization and Spherical Avoidance in 3-Dimensions

VII. Conclusion

Overall, we have shown that we were able to successfully reproduce a generalized robotic imitation learning policy from kinesthetic teaching using Dynamic Movement Primitives, Dynamic Time Warping, and Gaussian Mixture Regression to model preprocessed demonstrations. This robust policy from state to action mapping can guide robot behavior for applications across all industries. Kinesthetic teaching provides an intuitive medium to communicate and reproduce desired behavior.

Acknowledgment of AI

Code for this project was generated by LLMs:

- ChatGPT 5.2
- Claude Opus 4.5

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, May 2009, doi: 10.1016/j.robot.2008.10.024.

[2] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 15, 2002.

[3] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023, doi: 10.1177/02783649231201196.

[4] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, Feb. 2013, doi: 10.1162/NECO_a_00393.

[5] M. Müller, *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer, 2007, doi: 10.1007/978-3-540-74048-3.

[6] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent Service Robotics*, vol. 9, no. 1, pp. 1–29, 2016, doi: 10.1007/s11370-015-0187-9.

[7] A. Fabisch, "PyDMP: A simple implementation of dynamical movement primitives in Python," GitHub repository, 2024. [Online]. Available: https://github.com/AlexanderFabisch/PyDMP/tree/master. Accessed: Feb. 2026.

[8] Dynamic Time Warping Suite, "Welcome to the Dynamic Time Warp suite," 2026. [Online]. Available: https://dynamictimewarping.github.io/. Accessed: Feb. 2026.

[9] scikit-learn developers, "Gaussian Mixture Models — scikit-learn 1.5.0 documentation," scikit-learn.org, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/mixture.html. Accessed: Feb. 2026.

[10] J. Landers, 'https://github.com/JacktheLander/Lab-Projects/blob/main/Robot-Learning/Imitation_Learning/ImitationLearning.ipynb'