Skill Acquisition via Automated Multi-Coordinate Cost Balancing

Harish Ravichandar[1†], S. Reza Ahmadzadeh[2†], M. Asif Rana[1], and Sonia Chernova[1]

Abstract— We propose a learning framework, named Multi-Coordinate Cost Balancing (MCCB), to address the problem of acquiring point-to-point movement skills from demonstrations. MCCB encodes demonstrations simultaneously in multiple differential coordinates that specify local geometric properties. MCCB generates reproductions by solving a convex optimization problem with a multi-coordinate cost function and linear constraints on the reproductions, such as initial, target, and via points. Further, since the relative importance of each coordinate system in the cost function might be unknown for a given skill, MCCB learns optimal weighting factors that balance the cost function. We demonstrate the effectiveness of MCCB via detailed experiments conducted on one handwriting dataset and three complex skill datasets.

I. INTRODUCTION

The next generation of robots, that can operate in and adapt to unstructured and dynamic environments, must possess a diverse set of skills. However, it is implausible to pre-program robots with a library of all required skills. Learning from Demonstration (LfD) [1], [2] is a paradigm that aims to equip robots with the ability to learn efficiently from demonstrations provided by humans. Existing work in trajectory-based LfD has contributed a wide range of mathematical representations that encode skills from human demonstrations and then reproduce the learned skills at runtime. Proposed representations include Spring-damper systems with forcing functions [3], Gaussian Mixture Models (GMMs) [4]–[6], Neural Networks (NNs) [7], [8], Gaussian Processes (GPs) [9]–[11], and geometric objects [12], among others. Each of these representations is used to encode the demonstrations in a predefined space or coordinate system (e.g., Cartesian coordinates). In other words, a single best coordinate system for any given skill is assumed to both exist and be known. However, as we show in this work, the assumption that a single best coordinate system exists for each task does not hold. Further, encoding in only a single coordinate system prohibits the model from capturing some of the geometric features that underlie a demonstrated skill.

In this work, we contribute a learning framework that encodes demonstrations simultaneously in multiple coordinates, and balances the relative influences of the learned models in generating reproductions. The proposed framework, named Multi-Coordinate Cost Balancing (MCCB), encodes demonstrations in three differential coordinates: Cartesian, tangent, and Laplacian (Section III-A). Simultaneously learning in

† indicates equal contribution
[1] Georgia Inst. of Technology, Atlanta, GA. Email: {harish.ravichandar,asif.rana,chernova}@gatech.edu
[2] University of Massachusetts Lowell, Lowell, MA. Email: reza_ahmadzadeh@uml.edu

Fig. 1: A comparison of reproductions generated by considering different coordinates, illustrating the need for cost balancing.

these three coordinates allows our method to capture all of the underlying geometric properties that are central to a given skill. MCCB encodes the joint density of the time index and the demonstrations in each differential coordinate frame using a separate statistical model. Thus, given any time instant, we are able to readily obtain the conditional mean and covariance in each coordinate system (Section III-B). MCCB generates reproductions by solving an optimization problem with a blended cost function that consists of one term per coordinate. Each term penalizes deviations from the norm, weighted by the inverse of the expected variance in the corresponding coordinate system (Section III-C). Further, we subject the optimization problem to linear constraints on the reproductions, such as initial, target, and via point constraints. Our constrained optimization problem is convex with respect to the reproduction and hence can be solved efficiently.

A major hurdle in learning a wide variety of skills, without significant parameter tweaking, is that the relative importance of each differential coordinate (or the geometric feature) in encoding a given task is unknown ahead of time. For instance, consider the problem of encoding the demonstrations illustrated in Fig. 1. Using any one coordinate system in isolation, even when the most suitable one is known, does not yield good reproductions (the red, brown, and green dashed lines). To alleviate this problem, MCCB preferentially weights the costs defined in each coordinate (Fig. 2). Importantly, MCCB learns the optimal weights directly from the demonstrations without making task-dependent assumptions. To this end, MCCB solves a meta optimization problem that aims to minimize reproduction errors (Section III-D). As shown by the solid blue lines in Fig. 1, a cost function that optimally balances the costs in each coordinate yields better reproductions than any single-coordinate method.

Fig. 2: A flow diagram illustrating MCCB.

In summary, we contribute a unified task-independent learning framework that (1) encodes demonstrations simultaneously in multiple differential coordinates, (2) defines a blended cost function that incentivizes conformance to the norm in each coordinate system while considering expected variance, and (3) learns optimal weights directly from the demonstrations to balance the relative influence of each differential coordinate in generating reproductions. Further, MCCB is compatible with and complementary to several existing LfD methods that utilize different statistical representations and coordinate systems [10]–[16].

II. RELATED WORK

Learning from demonstration has attracted a lot of attention from researchers in the past few decades. While several categories of LfD methods exist [1], our work falls under the category of trajectory-based LfD. In this category, demonstrations take the form of trajectories and the methods aim to synthesize trajectories that accurately reproduce the demonstrations.

Dynamical systems-based trajectory learning methods, such as [5]–[7], encode demonstrations using statistical dynamical systems and generate reproductions by forward propagating the dynamics. While such deterministic methods exhibit several advantages, such as convergence guarantees and robustness to perturbations, they are restricted to learning in a single coordinate system and ignore inherent uncertainties in the demonstrations. They incentivize conformance to the norm even when demonstrations exhibit high variance.

Trajectory optimization methods, such as [17] and [18], focus on geometric features by minimizing costs specified using predefined norms. An optimization framework proposed in [19] attempts to adapt multiple demonstrations to new initial and target locations by minimizing the distance between the demonstrations and the reproduction according to a learned Hilbert space norm. Indeed, learning an appropriate Hilbert space norm is related to finding an appropriate coordinate system based on the demonstrations. However, similar to the dynamical systems-based methods, the methods in [17]–[19] are restricted to a single predefined or learned coordinate system and do not explicitly model and utilize the inherent time-dependent variations in the demonstrations.

Probabilistic trajectory-learning methods, such as [10], [11] and [14], on the other hand, capture and utilize the variation observed in the demonstrations. However, these methods are also restricted to encoding demonstrations in

a single predefined coordinate system that is assumed to be known.

Our design of the costs in each differential coordinate is inspired by the minimal intervention principle [13] that takes variance into account. While the approach in [13] does encode demonstrations in different frames of references, all the frames are restricted to Cartesian coordinates or orientation space. Furthermore, all the relevant frames for a given task are also expected to be provided by the user.

The motion planning framework in [15], complementary to our approach, utilizes a blended cost function, the construction of which is guided by probability distributions learned from the demonstrations. This framework incentivizes factors such as smoothness, manipulability, and obstacle avoidance, but is restricted to the Cartesian coordinate system. MCCB, on the other hand, encodes demonstrations in multiple differential coordinates and learns to optimally balance their relative influences, but does not consider factors such as manipulability and obstacle avoidance.

Differential coordinates have been extensively used in the computer graphics community [20], [21]. Prior work in trajectory learning that incorporates differential coordinates includes the Laplacian trajectory editing (LTE) algorithm [16]. Using Laplacian coordinates, the LTE algorithm adapts a single demonstration to new initial, target, and via points while preserving the shape. However, the LTE algorithm does not reason about the relative importances of multiple coordinates.

III. METHODOLOGY

The section describes the technical details of MCCB and its work flow as illustrated in Fig. 2.

A. Differential Coordinate Transformations

In this section, we define the differential coordinates and their corresponding transformations used in MCCB.

Cartesian: Let a discrete finite-length trajectory in n-dimensional *Cartesian coordinates* be denoted by $\boldsymbol{X} = [x(1) \ x(2) \cdots x(T)]^\top \in \mathbb{R}^{T \times n}$ and let $x(t) \in \mathbb{R}^n$ denote a discrete sample at time index t. This trajectory can be represented using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices representing the samples in the trajectory and $\mathcal{E}$ is the set of edges that represent the connections between the samples in the trajectory. The neighborhood $\mathcal{N}_t$ of each vertex $\mathcal{V}_t$ is defined by the set of adjacent vertices $\mathcal{V}_t'$. In the case of discrete-time trajectories, the edges between any given vertex and its two neighbors are assumed to carry unit weights, while all other edges carry zero weights.

Laplacian: It is known that the discrete Laplace-Beltrami operator for the trajectory $\boldsymbol{X}$ provides the *Laplacian coordinate* $\delta(t)$ as $\delta(t) \triangleq \sum_{t' \in \mathcal{N}_t} \frac{1}{\sum_{t' \in \mathcal{N}_t} 1}(x(t) - x(t'))$ [20]. Note that the above relationship can be written as a linear differential operator in matrix form

$$\boldsymbol{\Delta} = \boldsymbol{L}\boldsymbol{X} \qquad (1)$$

where $\boldsymbol{\Delta} = [\delta(1) \ \delta(2) \cdots \delta(T)]^\top \in \mathbb{R}^{T \times n}$ is the trajectory in the Laplacian coordinates, and $\boldsymbol{L} \in \mathbb{R}^{T \times T}$, called the

graph Laplacian, is given by

$$L = \begin{bmatrix} 1 & -1 & 0 & \dots & \dots & 0 \\ -0.5 & 1 & -0.5 & 0 & \dots & 0 \\ 0 & -0.5 & 1 & -0.5 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -0.5 & 1 & -0.5 \\ 0 & \dots & \dots & 0 & -1 & 1 \end{bmatrix} \quad (2)$$

As pointed out in [16], the Laplacian coordinates have meaningful geometric interpretations. Specifically, the Laplacian coordinates can be seen as the discrete approximations of the derivative of the unit tangent vectors of an arc-length parametrized continuous trajectory. In other words, the Laplacian coordinates measure the deviation of each sample from the centroid of its neighbors.

Tangent: While the Laplacian coordinates are discrete approximations of second order differential transformations, a discrete approximation of the first differential transformation is possible. Consider such a first order transformation using first order finite differences defined as $\gamma(t) \triangleq (x(t+1) - x(t))$, where $\gamma(t)$ is called the *tangent coordinate*. The matrix form of the above relationship results in a linear differential operator given by

$$\boldsymbol{\Gamma} = \boldsymbol{G}\boldsymbol{X} \quad (3)$$

where $\boldsymbol{\Gamma} = [\gamma(1)\ \gamma(2) \cdots \gamma(T)]^\top \in \mathbb{R}^{T \times n}$ is the trajectory in the tangent coordinates and $\boldsymbol{G} \in \mathbb{R}^{T \times T}$, called the graph incidence matrix, is given by

$$G = \begin{bmatrix} -1 & 1 & 0 & \dots & \dots & 0 \\ 0 & -1 & 1 & 0 & \dots & 0 \\ 0 & 0 & -1 & 1 & \dots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & -1 & 1 \\ 0 & \dots & \dots & 0 & 0 & -1 \end{bmatrix} \quad (4)$$

Similar to the Laplacian coordinates, the tangent coordinates have geometric interpretations. Specifically, the tangent coordinates can be seen as discrete approximations of the unnormalized tangent vectors of an arc-length parametrized continuous trajectory, i.e., the tangent coordinates measure the local direction of motion at each sample of the trajectory.

In our work, we assume that a set of N demonstrations in the Cartesian coordinates are available. Let the jth demonstration be denoted by $\boldsymbol{X}_d^j = [x_d^j(1)\ x_d^j(2) \cdots x_d^j(T)]^\top \in \mathbb{R}^{T \times n}$. Note that if the raw demonstrations are of varying duration in time, we perform time alignment using dynamic time warping. MCCB transforms each obtained demonstration $\boldsymbol{X}_d^j$ into a trajectory in the tangent coordinates (denoted by $\boldsymbol{\Gamma}_d^j$) and a trajectory in Laplacian coordinates (denoted by $\boldsymbol{\Delta}_d^j$) using (1) and (3), respectively.

B. Encoding in Multiple Differential Coordinates

This section defines the costs associated with each coordinate. With the demonstrations available in all three differential coordinates, we employ three independent Gaussian mixture models (GMMs)[1] to approximate the joint probability densities of time and the samples in each coordinate system.

[1]MCCB does not rely on the use of GMMs and any statistical representation that can provide the conditional estimates will suffice.

The GMM associated with the Cartesian coordinates attempts to approximate the joint density of t and x using a finite number of Gaussian basis functions as follows $\mathcal{P}(t,x;\theta_C) = \sum_{k=1}^{K_C} \mathcal{P}(k)\mathcal{P}(t,x|k)$, where K_C is the number of Gaussian basis functions, $\mathcal{P}(k) = \pi_C^k$ is the prior associated with the kth basis function, $\theta_C = \{\mu_C^1 \cdots \mu_C^{K_C}, \Sigma_C^1 \cdots \Sigma_C^{K_C}, \pi_C^1 \cdots \pi_C^{K_C}\}$ is the set of parameters of the GMM, and $\mathcal{P}(t,x|k)$ is the conditional probability density given by $\mathcal{P}(t,x|k) \sim \mathcal{N}\left(\begin{bmatrix} t \\ x \end{bmatrix}; \mu_C^k, \Sigma_C^k\right)$, where $\mu_C^k = \begin{bmatrix} \mu_t^k \\ \mu_x^k \end{bmatrix}$ is the mean and $\Sigma_C^k = \begin{bmatrix} \Sigma_t^k & \Sigma_{t,x}^k \\ \Sigma_{x,t}^k & \Sigma_x^k \end{bmatrix}$ is the covariance matrix of the kth Gaussian basis function.

We learn the parameters θ_C of the model using the Expectation-Maximization algorithm based on the demonstrations $\{\boldsymbol{X}_d^j\}_{j=1}^N$. Given the learned model and a time instant, the expected value of the conditional density $\mathcal{P}(x|t)$ is given by Gaussian mixture regression (GMR) [22] as follows

$$\hat{x}(t) = \mathbb{E}[x|t] = \sum_{k=1}^{K_C} h_C^k(t)(A_C^k t + b_C^k) \quad (5)$$

where $h_C^k(t) = \frac{\mathcal{P}(k)\mathcal{P}(t|k)}{\sum_{i=1}^{K_C} \mathcal{P}(i)\mathcal{P}(t|i)}$, $A_C^k = \Sigma_{x,t}^k(\Sigma_t^k)^{-1}$, $b^k = \mu_x^k + (t - \mu_t^k)$, and the conditional covariance is given by

$$\hat{\Sigma}_x(t) = Var[x|t] = \sum_{k=1}^{K_C} h_C^{k\,2}\,(\Sigma_x^k - \Sigma_{x,t}^k(\Sigma_t^k)^{-1}\Sigma_{t,x}) \quad (6)$$

Similar to the GMM learned in the Cartesian coordinates, we learn a second GMM in the tangent coordinates based on the demonstrations $\{\boldsymbol{\Gamma}_d^j\}_{j=1}^N$, and a third GMM in the Laplacian coordinates based on the demonstrations $\{\boldsymbol{\Delta}_d^j\}_{j=1}^N$. The expected values of the conditional densities $\mathcal{P}(\gamma|t)$ and $\mathcal{P}(\delta|t)$ are given by

$$\hat{\gamma}(t) = \mathbb{E}[\gamma|t] = \sum_{k=1}^{K_G} h_G^k(t)(A_G^k t + b_G^k) \quad (7)$$

$$\hat{\delta}(t) = \mathbb{E}[\delta|t] = \sum_{k=1}^{K_L} h_L^k(t)(A_L^k t + b_L^k) \quad (8)$$

and the corresponding conditional expectations are given by

$$\hat{\Sigma}_\gamma(t) = Var[\gamma|t] = \sum_{k=1}^{K_G} (h_G^k)^2\,(\Sigma_\gamma^k - \Sigma_{\gamma,t}^k(\Sigma_t^k)^{-1}\Sigma_{t,\gamma}) \quad (9)$$

$$\hat{\Sigma}_\delta(t) = Var[\delta|t] = \sum_{k=1}^{K_L} (h_L^k)^2\,(\Sigma_\delta^k - \Sigma_{\delta,t}^k(\Sigma_t^k)^{-1}\Sigma_{t,\delta}) \quad (10)$$

where the variables in (7)-(10) with subscripts G and L correspond to the tangent and Laplacian coordinates, respectively, and are defined similarly to the ones in (5)-(6).

C. Imitation via Optimization

In this section, we explain the design of our multi-coordinate cost function. MCCB generates reproductions by

solving a constrained optimization problem given by

$$\boldsymbol{X}_r = \arg\min_{\boldsymbol{X}} \; w_C J_C(\boldsymbol{X}) + w_G J_G(\boldsymbol{X})$$
$$+ w_L J_L(\boldsymbol{X}) \quad (11)$$
$$\text{s.t.} \qquad P_x \boldsymbol{X} = \boldsymbol{X}^* \quad (12)$$

where $\boldsymbol{X}_r \in \mathbb{R}^{T \times n}$ is the reproduction, $w_C,\ w_G,\ w_L \in \mathbb{R}^+$ are positive weights; $J_C,\ J_G,\ J_L : \mathbb{R}^{T \times n} \to \mathbb{R}^+$ are cost functions in the Cartesian, tangent, and Laplacian coordinates, respectively; $P_x \in \mathbb{R}^{m \times T}$ and $\boldsymbol{X}^* \in \mathbb{R}^{m \times n}$ define $m \in \mathbb{Z}^+$ linear constraints on $\boldsymbol{X}_r$. In practice, $m << n$ and we use the linear constraints to enforce constraints on initial, target, and via points.

We define the cost function in each coordinate system as follows

$$J_C(\boldsymbol{X}) = (\boldsymbol{X}(:) - \hat{\boldsymbol{X}}(:))^\top (\hat{\boldsymbol{\Sigma}}_{\boldsymbol{X}})^{-1} (\boldsymbol{X}(:) - \hat{\boldsymbol{X}}(:)) \quad (13)$$
$$J_G(\boldsymbol{X}) = (\boldsymbol{\Gamma}(:) - \hat{\boldsymbol{\Gamma}}(:))^\top (\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\Gamma}})^{-1} (\boldsymbol{\Gamma}(:) - \hat{\boldsymbol{\Gamma}}(:)) \quad (14)$$
$$J_L(\boldsymbol{X}) = (\boldsymbol{\Delta}(:) - \hat{\boldsymbol{\Delta}}(:))^\top (\hat{\boldsymbol{\Sigma}}_{\boldsymbol{\Delta}})^{-1} (\boldsymbol{\Delta}(:) - \hat{\boldsymbol{\Delta}}(:)) \quad (15)$$

where $\hat{\boldsymbol{\Sigma}}_{\boldsymbol{X}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{\Gamma}}, \hat{\boldsymbol{\Sigma}}_{\boldsymbol{\Delta}} \in \mathbb{R}^{nT \times nT}$ denote the block diagonal matrices formed with the conditional covariances $\hat{\Sigma}_x(t), \hat{\Sigma}_\gamma(t),$ and $\hat{\Sigma}_\delta(t)$, respectively, for all values of t. Further, the notation $(:)$ denotes vectorization - for instance, $\boldsymbol{X}(:), \hat{\boldsymbol{X}}(:) \in \mathbb{R}^{nT}$ denote the vectorized trajectories formed by vertically stacking $x(t)$ and $\hat{x}(t)$ for all values of t, respectively. Note that we construct the trajectories Γ and Δ in (14) and (15) from $\boldsymbol{X}$ via the linear operators defined in (3) and (1), respectively. MCCB penalizes deviations from the conditional mean in each coordinate system. However, deviations are penalized less (more) severely if high (low) variance is observed in the demonstrations at any given time.

D. Automated Cost Balancing

In order to obtain reproductions that successfully imitate demonstrations of a wide variety of skills, the weights w_C, w_G, and w_L have to be chosen with care. Indeed, they preferentially weight the costs defined in each differential coordinate and thereby manipulate the relative incentive for successful imitation in each coordinate system.

We learn these weights directly from the available demonstrations. Note that, for known weights, the constrained optimization problem in (11) is convex in $\boldsymbol{X}$. We estimate the weights in the following form

$$\hat{w}_C = \frac{\alpha_C}{\beta_C}; \quad \hat{w}_G = \frac{\alpha_G}{\beta_G}; \quad \hat{w}_L = \frac{\alpha_L}{\beta_L} \quad (16)$$

where $\beta_C,\ \beta_G,\ \beta_L \in (0, 1]$, such that $\sum_i \beta_i = 1$, are positive scaling factors used to correct for inherent differences in the magnitudes of the costs, and $\alpha_C,\ \alpha_G,\ \alpha_L \in [0, 1]$, such that $\sum_i \alpha_i = 1$, are positive weights used to preferentially weight the cost defined in each coordinate system. MCCB estimates the scaling factors β_i's as follows

$$\beta_i = \frac{\sum_{j=1}^N J_i(\boldsymbol{X}_d^j)}{\sum_l \sum_{j=1}^N J_l(\boldsymbol{X}_d^j)}, \quad \forall i, l = \{C, G, L\} \quad (17)$$

With the scaling factors compensating the inherent scale difference in the costs, we compute the preferential weighting

factors α_i's that minimize reproduction error. To this end, we formulate the following meta optimization problem

$$\{\alpha_C, \alpha_G, \alpha_L\} = \arg\min_{\alpha_C, \alpha_G, \alpha_L} \sum_{j=1}^N \text{SSE}(\boldsymbol{X}_r^j, \boldsymbol{X}_d^j) \quad (18)$$
$$\text{s.t.} \sum_i \alpha_i = 1, \; \forall i = \{C, G, L\} \quad (19)$$

where $\text{SSE}(\cdot)$ denotes the sum of squared errors computed over time, and $\boldsymbol{X}_r^j$ is the solution to the following optimization problem

$$\boldsymbol{X}_r^j = \arg\min_{\boldsymbol{X}} \; \left(\frac{\alpha_C}{\beta_C}\right) J_C(\boldsymbol{X}) + \left(\frac{\alpha_G}{\beta_G}\right) J_G(\boldsymbol{X})$$
$$+ \left(\frac{\alpha_L}{\beta_L}\right) J_L(\boldsymbol{X}) \quad (20)$$
$$\text{s.t.} \qquad P_x \boldsymbol{X} = \boldsymbol{X}_j^* \quad (21)$$

where $P_x \boldsymbol{X} = \boldsymbol{X}_j^*$ denotes specific linear constraints pertaining to the demonstration $\boldsymbol{X}_d^j$, such as initial, target, and via points. Solving the above meta-optimization problem results in the preferential weights α_i's that minimize reproduction errors of the solutions generated by the original constrained optimization problem in (11)-(12).

IV. EXPERIMENTAL EVALUATION

This section describes the design and discusses the results of four experiments conducted to evaluate MCCB. In each experiment, we compared the performances of the following approaches:

1) *Cartesian-coordinates*: $w_C = 1$, $w_G = 0$, $w_L = 0$
2) *Tangent-coordinates*: $w_C = 0$, $w_G = 1$, $w_L = 0$
3) *Laplacian-coordinates*: $w_C = 0$, $w_G = 0$, $w_L = 1$
4) *Uniform weighting*: $w_C = 1/3$, $w_G = 1/3$, $w_L = 1/3$
5) *MCCB*: $w_C = \hat{w}_C$, $w_G = \hat{w}_G$, $w_L = \hat{w}_L$

We measured the performance of each approach by the following geometric and kinematic metrics: *Swept Error Area (SEA)* [23], *Sum of Squared Errors (SSE)*, *Dynamic Time Warping Distance (DTWD)*, and *Frechet Distance (FD)* [24]. These metrics allow us to evaluate different aspects of each method's performance. The SEA and SSE metrics penalize both spatial and temporal misalignment, and thus evaluate kinematic performance. On the other hand, the DTWD and FD metrics penalize spatial misalignment while disregarding time misalignment, and thus evaluate geometric performance. Further, the SEA, SSE, and DTWD metrics evaluate aggregate performance by summing over or averaging across all the samples of each reproduction. The FD metric, on the other hand, computes the shortest possible cord length required to connect the demonstration and the reproduction in space while allowing time re-parametrization of either trajectory, and thus measures maximal deviation in space. Note that the SEA metric is restricted to 2-dimensional data, so we only report it for one of our experiments.

In all the experiments, we used the position constraints in (12) to enforce both initial and end point constraints uniformly across all the methods being compared. Further,

Fig. 3: Qualitative performance of MCCB on the LASA hand-writing dataset. Demonstration (gray), reproductions (blue), and expected mean position (dashed red) are shown.

Fig. 4: Box plots, with mean (brown star) and median (red line), illustrate the performance of each approach on the handwriting task.

we uniformly set the number of Gaussian basis functions to five across all the coordinates and all the experiments.

A. Handwriting Skill

This experiment evaluates MCCB on the publicly available LASA human handwriting library [5], that consists of handwriting motions collected from pen input using a Tablet PC. The library contains a total of 25 handwriting motions, each with 7 demonstrations.

Fig. 3 shows that MCCB yields reproductions that are qualitatively similar to the demonstrations while satisfying the end-point constraints across all motions. As shown in Fig. 4, quantitative analysis indicates that MCCB ($\bar{\alpha}_C = 0.1814$, $\bar{\alpha}_G = 0.4958$, $\bar{\alpha}_L = 0.3228$)[2] and three of the four baselines performed comparably with respect to the SEA, FD, SSE, and DTWD metrics, while the Cartesian

[2]Weighting factors averaged over all 25 skills in the LASA dataset

Fig. 5: Snapshots illustrating the experimental setup for the picking (left), pressing (center), and pushing (right) skills.

baseline performed poorly in comparison. This is consistent with the fact that the demonstrations within the LASA dataset emphasize strong similarities in shape.

B. Picking Skill

The second experiment evaluates the performance of MCCB in a picking task (Fig. 5). The data consists of six kinesthetic demonstrations, each a 3-dimensional robot end-effector position trajectory recorded as a human guided the robot in picking up two magnets atop two blocks. We enforced two via-point constraints (one at each picking point) in addition to the end-point constraints.

As shown in Fig. 6(a), MCCB generated reproductions that are qualitatively similar to the demonstrations while satisfying all the position constraints. Quantitative evaluations reveal that learning in tangent coordinates yielded better reproductions than learning in Cartesian and Laplacian coordinates (Fig. 7). This was expected since the demonstrations of this task, much like the LASA dataset, emphasize shape similarity. Further, MCCB ($\alpha_C = 0.2362$, $\alpha_G = 0.5451$, $\alpha_L = 0.2187$) yielded the best performance, with respect to all three metrics. In fact, uniform weighting yielded poorer results, with respect to all three metrics, than when considering only the tangent coordinates. The results of this experiment show that while multi-coordinate methods can yield strong performance, it is critical that we balance the weights appropriately.

C. Pressing Skill

In this experiment, we evaluated MCCB's ability to learn pressing skills (Fig. 5). The data consists of six kinesthetic demonstrations, each a 3-dimensional robot end-effector position trajectory recorded as a human guided the robot in pressing two cylindrical pegs into their respective holes.

As shown in Fig. 6(b), MCCB successfully reproduced the demonstrations. Note that MCCB is capable of automatically capturing and reproducing the consistencies across the demonstrations in certain regions without any position constraints. Fig. 8 illustrates the performance of MCCB and the baselines with respect to three different metrics. Learning in Cartesian coordinates resulted in the better performance compared to learning in tangent and Laplacian coordinates. Quantitative evaluations further demonstrate that MCCB ($\alpha_C = 0.6735$, $\alpha_G = 0.2034$, $\alpha_L = 0.1231$) consistently yielded the best performance with respect to all three metrics. The results of this experiment, in light of the results in Section IV-B, suggest that the relative importance of each of the differential coordinates vary across different skills.

Fig. 6: Qualitative performance of MCCB on the picking, pressing, and pushing datasets. Demonstration (gray), reproductions (blue), expected mean position (dashed red), initial (black squares), and target (black stars) are shown.

Fig. 7: Box plots, with mean (brown star) and median (red line), illustrate the performance of each approach on the picking dataset.

Fig. 8: Box plots, with mean (brown star) and median (red line), illustrate the performance of each approach on the pressing dataset.

D. Pushing Skill

The final experiment evaluates the performance of MCCB in a pushing task (Fig. 5). The data consists of six kinesthetic demonstrations, each a 3-dimensional robot end-effector position trajectory recorded as a human guided the robot in sliding closed the lid of a wooden box.

As shown in Fig. 6(c), MCCB successfully generated reproductions that are similar to the demonstrations. As evidenced by quantitative evaluations in Fig. 9, encoding demonstrations in the Laplacian coordinates yielded better performance, with respect to all three metrics, when compared to learning only in either of the other two coordinates, while, MCCB ($\alpha_C = 0.0123$, $\alpha_G = 0.045$, $\alpha_L = 0.9427$) consistently outperformed all the other approaches. Note that learning in the Laplacian coordinates alone resulted in better performance than uniformly weighting of all the coordinates. These results are consistent with the results from the previous sections and indicate that MCCB yields consistently good performance. The results are summarized in Table I.

Fig. 9: Box plots, with mean (brown star) and median (red line), illustrate the performance of each approach on the pushing dataset.

	Single Coordinate			Multi-Coordinate	
	Cartesian	Tangent	Laplacian	Uniform W.	MCCB
Handwriting		✓ ✓	✓ ✓	✓	✓
Picking		✓			✓
Pressing	✓				✓
Pushing			✓		✓

TABLE I: Orange check marks denote the most relevant coordinate and green check marks denote the best performing method.

V. DISCUSSION AND CONCLUSION

We introduced MCCB, a learning framework for encoding demonstrations in multiple differential coordinates, and automated balancing of costs defined in those coordinates. As shown in Table I, we demonstrated that the relative effectiveness of each coordinate system is not consistent across a variety of tasks since any given skill might be better suited for learning in one (or more) coordinate system(s). Furthermore, uniform weighting of costs in different coordinates does not consistently yield the best results across different skills. Indeed, uniform weighting, in some cases, yielded poorer performances compared to when only one coordinate system was used. On the other hand, MCCB learned to balance the costs and consistently yielded the best performance. Since the weights are learned directly from the demonstrations, MCCB makes no task-specific assumptions and does not require tedious parameter tuning. Note that although we used GMMs as the base representation in this work, MCCB is agnostic to the statistical model used to encode the demonstrations in each coordinate system, and thus can be combined with other techniques, such as [10]–[16]. Furthermore, MCCB can be extended to include more coordinate systems that capture additional trajectory features.

ACKNOWLEDGMENT

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, 2009.

[2] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer handbook of robotics*. Springer, 2008, pp. 1371–1394.

[3] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 763–768.

[4] 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, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[5] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[6] H. C. Ravichandar and A. Dani, "Learning position and orientation dynamics from demonstrations via contraction analysis," *Autonomous Robots*, pp. 1–16, 2018.

[7] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven lyapunov candidates," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1216–1222.

[8] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *International Conference on Machine Learning*, 2014, pp. 829–837.

[9] M. Schneider and W. Ertel, "Robot learning by demonstration with local gaussian process regression," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 255–260.

[10] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, and B. Boots, "Towards robust skill generalization: Unifying learning from demonstration and motion planning," in *Proceedings of the 2017 Conference on Robot Learning (CoRL)*, 2017.

[11] J. Umlauft and S. Hirche, "Learning stable stochastic nonlinear dynamical systems," in *International Conference on Machine Learning*, 2017, pp. 3502–3510.

[12] S. R. Ahmadzadeh, M. A. Rana, and S. Chernova, "Generalized cylinders for learning, reproduction, generalization, and refinement of robot skills." in *Robotics: Science and Systems*, 2017.

[13] S. Calinon, D. Bruno, and D. G. Caldwell, "A task-parameterized probabilistic model with minimal intervention control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3339–3344.

[14] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in neural information processing systems*, 2013, pp. 2616–2624.

[15] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 819–826, 2017.

[16] T. Nierhoff, S. Hirche, and Y. Nakamura, "Spatial adaption of robot trajectories based on laplacian trajectory editing," *Autonomous Robots*, vol. 40, no. 1, pp. 159–173, 2016.

[17] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 489–494.

[18] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[19] A. D. Dragan, K. Muelling, J. A. Bagnell, and S. S. Srinivasa, "Movement primitives via optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2339–2346.

[20] Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, and H.-P. Seidel, "Differential coordinates for interactive mesh editing," in *Shape Modeling Applications*. IEEE, 2004, pp. 181–190.

[21] B. Lévy, "Laplace-beltrami eigenfunctions towards an algorithm that" understands" geometry," in *IEEE International Conference on Shape Modeling and Applications*, 2006, pp. 13–13.

[22] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.

[23] S. M. Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[24] M. M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, vol. 22, no. 1, pp. 1–72, 1906.