2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Learning Motion Trajectories from Phase Space Analysis of the Demonstration

Paul Gesel[1], Momotaz Begum[1], Dain La Roche[2]

***Abstract*— A major goal of learning from demonstration is task generalization via observation of a teacher. In this paper, we propose a novel framework for learning motion from a single demonstration. Our approach reconstructs the demonstrated trajectory's phase space curve via a linear piece-wise regression method. We approximate dynamics of trajectory segments with linear time invariant equations, each yielding closed form solutions. We show convergence to desired phase space states via an energy-based analysis. The robustness of the model is evaluated on a robot for a sequential trajectory task. Additionally, we show the advantages that the phase space model has over the dynamic motion primitive for a kinematic based task.**

I. Introduction

Learning motion trajectories from demonstration data is probably the most matured area in robotics research on learning from demonstration (LfD) [5], [3]. The major goal of trajectory LfD is robust generalization: encoding motion data in such a way that a similar motion can be generated in new contexts. A robust trajectory learning algorithm should have the following properties: 1) be able to learn from few demonstrations, 2) be robust against spatial perturbation (e.g change in goal and/or starting position, obstacle avoidance) and temporal perturbations (e.g. an unplanned obstacle prevents reaching the goal position [9]), and 3) be able to satisfy various kinematic constraints of the demonstrated trajectory (e.g. certain velocities need to be achieved at certain positions). These properties enable a model to adapt learned trajectories to different environmental contexts, while preserving the key characteristics of the demonstrated motion. A flurry of research has been done on learning complex motions from human demonstrations [1], [15], [14], [2]. None of the existing trajectory LfD algorithms, however, exhibits all three properties mentioned above.

This paper proposes a novel approach of motion learning from a single demonstration that we name the phase space model (PSM). The PSM approximates second order trajectory dynamics via a linear piece-wise method. The core of the PSM is the phase space transition function (PSTF) that, given an initial phase space state (position, velocity), always converges to a specified phase space state. This enables the PSM to learn various kinematic constraints in the demonstrated trajectory. We develop and implement the necessary conditions to ensure that the PSTF converges to the desired phase space state. Through a piece-wise combination of PSTFs, an approximation of the demonstration is reproduced. We expand the PSM to multiple dimensions and propose a method to achieve trajectory synchronization. The PSM is inherently a time-independent method (since all analysis are performed in the phase space) capable of dealing with spatial and temporal perturbations.

[1] Computer Science Department, University of New Hampshire, Durham, USA {pgesel,mbegum}@cs.unh.edu [2] Department of Kinesiology, University of New Hampshire, Durham, USA {Dain.LaRoche}@unh.edu

II. Background

Two methods of encoding low-level motion data have gained tremendous popularity in the LfD literature: a probabilistic approach [17] and a dynamic system approach [7], [8].

The probabilistic approach of trajectory learning in [17] creates a statistical representation of the demonstrated motion trajectories through Gaussian-mixture modeling (GMM). This approach learns a time-dependent trajectory distribution from several demonstrations of the task. The model and its variants have been used in teaching skills to robots, such as goal-directed pick-n-place actions [17], [4] and pouring from cups [2]. The common criticisms of GMM-based trajectory learning are: the requirement for multiple demonstrations, the inability to adapt to different goal positions and avoid obstacles online, and the inherent time-dependence. Being a time-dependent approach, GMM-based trajectory learning requires a heuristic to re-index time to generate a new trajectory in the presence of temporal perturbations [9]. Stable Estimator of Dynamical Systems (SEDS) is dynamic system theory variant of the GMM-based trajectory learning that demonstrates robustness in temporal and spatial perturbations [10]. However, in the SEDS framework, learning a new motion requires solving a constrained optimization problem. Additionally, like GMM-based trajectory learning, SEDS requires multiple trajectory demonstrations.

The dynamic motion primitive (DMP) exploits the stable point-attractor dynamics of a spring-mass system to encode motion data [8]. The stable dynamics naturally enable generalization in new contexts, such as on-line trajectory adaptation and obstacle avoidance [7], [8], [18]. DMPs have been used to teach various skills to LfD-powered robots, e.g. locomotion [13], T-ball batting [16], playing tic-tac-toe [14], and drumming [6]. DMPs can encode joint trajectory motion from a single demonstration. The phase variable, however, in the DMP is a function of time, which implicitly makes DMP a time dependent approach. Accordingly, the issue of temporal re-indexing, as discussed earlier in this section, occurs when reproducing trajectories learned with DMPs. Another critical limitation of DMP in learning trajectories is that DMP only considers the temporal constraint of the

978-1-5386-6027-0/19/$31.00 ©2019 IEEE

task (i.e., it tries to reach the goal within a specified time). There are many tasks, however, that require other kinematic constraints to be satisfied. For example, learning to throw a ball a specific distance requires a specific velocity and position combination. Modifications of the original DMP formulations have been proposed to model such behavior [12], [11]. These modifications, however, will fail if a trajectory needs to execute multiple instances of such behavior.

The PSM proposed in this paper overcomes several shortcomings of existing trajectory LfD algorithms. The PSM learns trajectories from only one demonstration. Goals in the proposed PSM are defined in the phase space and the PSTF ensures stable transition between any two phase space states. This enables the model to inherently learn kinematic constraints in the demonstrated trajectory. Finally, the model is able to deal with spatial and temporal perturbations (e.g. change in start position, online obstacle avoidance, and goal adaptation).

This paper describes the theoretical foundation of the PSM model along with validation results for two tasks: learning to feed a person and learning to roll a cylinder a certain distance.

III. The Proposed Approach

The proposed PSM employs linear time invariant systems to model segmented trajectory dynamics via a phase space analysis.

A. Fundamental of phase space

Phase space is an n-dimensional manifold that represents all possible states of a dynamical system. Each point in the phase space indicates the state of the system at a specific time. Tracing the time evolution of the state variables from an initial state generates a phase space curve. The direction traveled in phase space at any given state is determined by the velocity function. $d\vec{\eta}\ /dt = V\ (\vec{\eta}) = (dx/dt, dv/dt)$ The velocity function is a vector field in the phase space that is defined for all possible states.

Fig. 1 shows a phase space representation of two dynamic systems: an under damped and a critically damped spring-mass system. Both are models of second order mechanical systems. In this case, a three dimension phase space is plotted

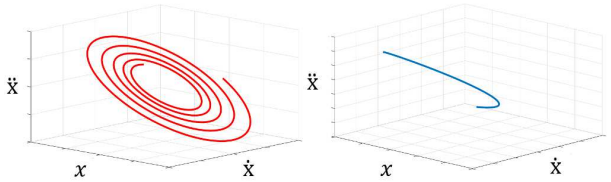

Fig. 1: Phasemap of two mechanical dynamical systems. On the left is under damped and on the right is critically damped

with position, velocity, and acceleration. The position and velocity are independent state variables, while the acceleration is depended. Thus, a second order dynamic system of n independent variables can be fully represented in a $(n+1)$ dimensional phase space.

B. Phase space transition function

We define a PSTF as a dynamic system that reaches a desired phase space state (position, velocity), given an initial state. We exploit the properties of conservative vector fields to develop such a function. A Conservative dynamic system is a special type of system that conserves certain properties along its phase space curve, e.g. second order mechanical systems, without damping, conserve energy. A potential function exists for these types of systems,

$$\nabla f = F$$

where f is the system's potential function and F is a conservative vector field. Given a potential function, the gradient theorem shows that the work input from the vector field is path independent.

$$W = \int_a^b F \cdot ds = \int_a^b \nabla f \cdot ds = f(b) - f(a) \qquad (1)$$

The work, W, is the system's energy input. For a one dimensional unit mass point particle, $(m = 1)$, the energy input is stored in the form of kinetic energy (KE) according to equation (2).

$$KE = \frac{1}{2}\dot{x}^2 \qquad (2)$$

In equation (1), F could be any conservative vector field. We chose a linear model in the following form,

$$\ddot{x} = kx + c \qquad (3)$$

where k, and c are constants. Equation (1) is evaluated with $F = kx + c$ over the interval $[x_c, x_n]$ to find the kinetic energy input.

$$KE = k\frac{(x_n^2 - x_c^2)}{2} + c\,(x_n - x_c) + \frac{\dot{x}_c^2}{2} \qquad (4)$$

Here, $\dot{x}_c$ is the current velocity, x_c is the current position, and $(\dot{x}_n, x_n)$ is the next phase space state to be reached. We chose a value for c such that the velocity boundary condition $\dot{x}_n$ is met at $x = x_n$.

$$c = \frac{\dot{x}_n^2 - \dot{x}_c^2}{2\,(x_n - x_c)} + k\frac{x_n^2 - x_c^2}{2\,(x_n - x_c)} \qquad (5)$$

Since equation (3) is a linear second order system, a closed form solution exists. Applying the boundary condition that $x = x_c$ and $\dot{x} = \dot{x}_c$ at $t = 0$, generates the following closed form solution.

$$x = \left(\frac{x_c + \frac{\dot{x}_c}{\sqrt{k}} + \frac{c}{k}}{2}\right) e^{\sqrt{k}t} + \left(\frac{\frac{-\dot{x}_c}{\sqrt{k}} + x_c + \frac{c}{k}}{2}\right) e^{-\sqrt{k}t} - \frac{c}{k} \qquad (6)$$

It should be noted that equation (6) meets boundary conditions only if the $\dot{x}_n$ has the same sign as $(x_n - x_c)$ or is zero. It doesn't make sense to reach x_n coming from x_c with a velocity in the direction of x_c.

Equation (6) meets starting and ending boundary conditions, while maintaining k as a free parameter. For this reason, we denote equation (3) as a PSTF. An appropriate value of k in a PSTF can produce trajectories that approximate the

dynamics of a demonstration. For example, fig. 2 shows how different values of k generate a wide variety of trajectories. In fig. 2, each trajectory represents a unique PSTF.

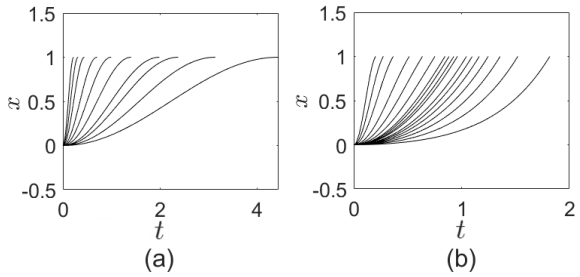

Fig. 2: Trajectories for different values of k in equation (6) with $x_c = 0$, $x_n = 1$ and (a) $\dot{x}_c = 0$, $\dot{x}_n = 0$. (b) $\dot{x}_c = 0$, $\dot{x}_n = 1$.

C. Piece-wise PSTFs

The approach of the PSM is to break the phase space into segments and then approximate each segment with a PSTF. The first and second numerical derivatives of the demonstration are required to reconstruct its phase space curve. It is difficult to get an accurate derivative approximation for noisy data, consequently, we apply a Gaussian kernel smoothing to the demonstration before modeling it. The projection of the three-dimensional phase space onto the position-acceleration plane can then be modeled with piece-wise PSTFs.

To clarify the piece-wise behavior, equation (3) can be re-written as follows,

$$\ddot{x} = k_n x + c_n \tag{7}$$

where n is the segment index. Fig. 3 shows the acceleration of a reaching motion vs. the position. The reaching motion is segmented into three spatial intervals: $[0, .3]$, $[.3, .8]$, and $[.8, 1]$. On the first interval $[0, .3]$, k_1 is evaluated in equation (7) with the $x_1 = .3$ and $\dot{x}_1 = 1.6$. Once the phase space state (x_1, $\dot{x}_1$) is reached, the second segment on the interval $[.3, .8]$ is evaluated with k_2. The process is repeated for each PSTF until the trajectory is complete. In general, a trajectory segmented into N pieces has N values of k and $N + 1$ phase space states (x, $\dot{x}$). The demonstrated trajectory can be reproduced via evaluating the value of k that corresponds to the current spatial interval in equation 7. To allow for online goal adaptation, the spatial intervals are expressed relative to the goal's location. Changing the goal location during the reaching motion only changes x and c_n. The change in goal position may also cause the current spatial interval being evaluated to change, hence, changing k_n as well.

For this paper, we chose the number of segments (N) manually and uniformly distributed cut points in time. One PSTF is required for each time the velocity crosses zero, hence, the lower bound on N is equal to the number of times the demonstrated trajectory changes direction. In general, the accuracy of the trajectory reproduced by the PSM increases as the number of segments increase.

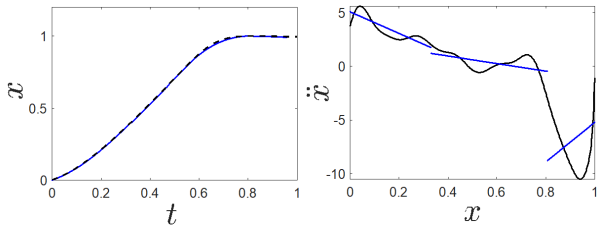

Fig. 3: The reaching motion demonstration is shown in black and the model in blue. On the left is the demonstrated trajectory and the PSM approximation. On the right is the acceleration-position relationship and the PSM's linear piece-wise approximation.

D. Stability

We term the stability of a PSFT as its property to reach a desired velocity (e.g $\dot{x}_n$) at a desired position (e.g. x_n) from an initial phase space state. Equation (7) does not generate a stable trajectory for all values of k_n. The stability depends on both the initial velocity and k_n. Through an energy-based analysis, we determine the necessary condition to achieve stability. If each individual PSTF is stable, then we can consider the entire trajectory as stable.

Fig. 4 shows the kinetic energy of three trajectories on the interval [0,1]. These three cases need to be considered: the blue curve with $k < 0$ and $min(KE) > 0$, the orange curve with $k > 0$ and $min(KE) = 0$, and the red curve with $k > 0$ and $min(KE) < 0$. The blue curve is stable, the red curve is unstable, and the orange curve is an unstable equilibrium. The resulting curves were generated with equation (4). Any curve where $k < 0$ is periodic. The amplitude corresponds to the difference between the two intersection points of the KE curve with zero, hence, $k < 0$ is a stable trajectory. This is shown by evaluating $k < 0$ in equation (6), which results in a sinusoid with a constant offset. If $k > 0$, the result is an exponential function and inherently unstable. Locations where the KE curve intersect zero represent turnaround points, since kinetic energy can't be negative, e.g. the red curve will stop at 0.25 and then accelerates toward negative infinity. The orange curve intersects zero with a slope of zero, which results in an unstable equilibrium point. The necessary condition to ensure stability is that the minimum KE on the interval $[x_c, x_n]$ must be greater than zero.

Since the KE curves are quadratic, only one minimum exists and it can be solved for analytically. The derivative of the KE with respect to x is set equal to zero and solved to find x_{min},

$$k_n x_{min} + c_n = 0$$

$$x_{min} = -c_n / k_n \tag{8}$$

where x_{min} is the value of x that corresponds to the minimum kinetic energy. If x_{min} is on the interval $[x_c, x_n]$, then the value of k_n needs to be adjusted, such that, $KE(x_{min}) > 0$. A constraint on k_n is determined by evaluating equation (4) at x_{min}.

$$KE(x_{min}) > 0$$

$$k_n \frac{\left((-c_n/k_n)^2 - x_c^2\right)}{2} + c_n\left(-c_n/k_n - x_c\right) + \frac{\dot{x}_c^2}{2} > 0 \quad (9)$$

Equation (9) is quadratic and can be solved to produce an upper boundary on k, denoted k_{max}.

$$k_{max} < \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$

$$A = \frac{x_n^2 - x_c^2}{(x_n - x_c)} x_c - x_c^2 - \left(\frac{x_n^2 - x_c^2}{2\,(x_n - x_c)}\right)^2$$

$$B = \left(\frac{x_n^2 - x_c^2}{(x_n - x_c)} \frac{\dot{x}_n^2 - \dot{x}_c^2}{(x_n - x_c)} - 2\frac{\dot{x}_n^2 - \dot{x}_c^2}{(x_n - x_c)} x_n + \dot{x}_c^2\right)$$

$$C = -\left(\frac{\dot{x}_n^2 - \dot{x}_c^2}{(x_n - x_c)}\right)^2$$

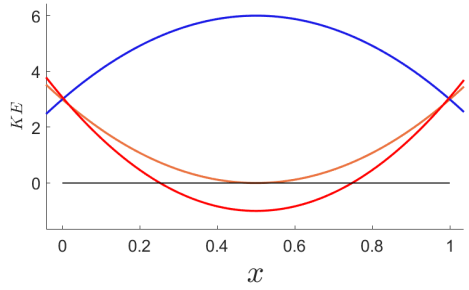

Fig. 4: Kinetic energy vs. position for three trajectories from equation (4). The following are the parameters for each curve: the blue curve: $k = -24$, $\frac{\dot{x}_c^2}{2} = 3$, the orange curve: $k = 24$, $\frac{\dot{x}_c^2}{2} = 3$, and the red curve: $k = 32$, $\frac{\dot{x}_c^2}{2} = 3$

The analysis thus far has assumed that the initial velocity is in the direction of x_n, however, this may not always be the case in the presence of perturbations, e.g. a person physically pushes the robot arm away from the goal position. The orange curve in fig. 4 starts at a kinetic energy of 3, which can have either positive or negative velocity. If the velocity is away from x_n, then it will approach negative infinity.

One additional stability constraint is applied to ensure the PSTF reaches the phase space state (x_n, $\dot{x}_n$) in the case that the $\dot{x}_c$ in the wrong direction. Equation (7) needs to be adjusted to equation (10).

$$\ddot{x} = \begin{cases} k_n x + c_n & \dot{x} \geq \frac{x_n - x_c}{|x_n - x_c|} \\ m & \dot{x} < \frac{x_n - x_c}{|x_n - x_c|} \end{cases} \quad (10)$$

In equation (10), m is a positive constant. If the velocity is in the direction away from x_n, a constant acceleration is applied, until the velocity reaches zero. The value of m influences how much overshoot in the negative direction the trajectory will have and can be tuned to the specific robot's dynamics.

E. Synchronizing multiple dimensions

One approach to achieve multidimensional trajectories is to model each dimension separately with piece-wise PSTFs. The problem occurs when one of the dimensions experience perturbations, causing the trajectory to become out of sync. A grasping task requires that the robot gripper comes in at a certain angle. If the Cartesian dimensions x, y, and z get out of sync, the robot gripper will approach the goal at the wrong angle. Despite its time-independent nature, the PSM can deal with situations that require timing constraints.

Equation (6) can be re-written as a function of its derivative and then inverted to solve for t as a function of x.

$$x = \frac{\dot{x}}{\sqrt{k}} + 2\left(\frac{\frac{-\dot{x}_c}{\sqrt{k}} + x_c + \frac{c}{k}}{2}\right) e^{-\sqrt{k}t} - \frac{c}{k}$$

$$t = \frac{\log\left(\frac{-\dot{x}_c}{\sqrt{k}} + x_c + \frac{c}{k}\right)}{\sqrt{k}} - \frac{\log\left(x - \frac{\dot{x}}{\sqrt{k}} + \frac{c}{k}\right)}{\sqrt{k}} \quad (11)$$

The time that the nth segment takes to complete can be calculated by substituting x_n for x, c_n for c, and k_n for k in equation (11).

$$t_n = \frac{\log\left(\frac{-\dot{x}_c}{\sqrt{k_n}} + x_c + \frac{c_n}{k_k}\right)}{\sqrt{k_n}} - \frac{\log\left(x_n - \frac{\dot{x}_n}{\sqrt{k_n}} + \frac{c_n}{k_n}\right)}{\sqrt{k_n}} \quad (12)$$

The PSM uses a piece-wise combination of equation (10), consequently, the total time to trajectory completion is the sum of the time it takes to complete each segment as shown in equation (13).

$$t_e = \sum_{n=0}^{N} t_n \quad (13)$$

Each dimension has its own t_e, which quantifies the progression through each dimension. To synchronize each dimension, trajectories with smaller t_e values should be slowed down to let the others catch up. We denote the largest t_e as t_E and adjust equation (10) again.

$$\ddot{x} = \begin{cases} k_n x + c_n - T\dot{x}(t_E - t_e) & \dot{x} \geq \frac{x_n - x_c}{|x_n - x_c|} \\ m & \dot{x} < \frac{x_n - x_c}{|x_n - x_c|} \end{cases} \quad (14)$$

T is essentially a breaking term and is handpicked in our implementation. In general, it needs to be large enough, such that, the trajectories closer to the end of the movement slow down enough for the ones further from the end to catch up. The $T\dot{x}(t_E - t_e)$ term should be on the magnitude of $k_n x + c_n$ term to have a significant impact. Fig. 5 shows a three-dimensional trajectory with and without the time synchronization applied. Notice that the x component starts with a lower t_e than both y and z. With the time synchronization included, the x component waits for the other components to approach 2.5 seconds until its start moving at its original rate.

Another added benefit to having $t_e = f(x, \dot{x})$ is the ability to couple different dimensions. This is particularly useful for grasping task. A time-based function approximation can

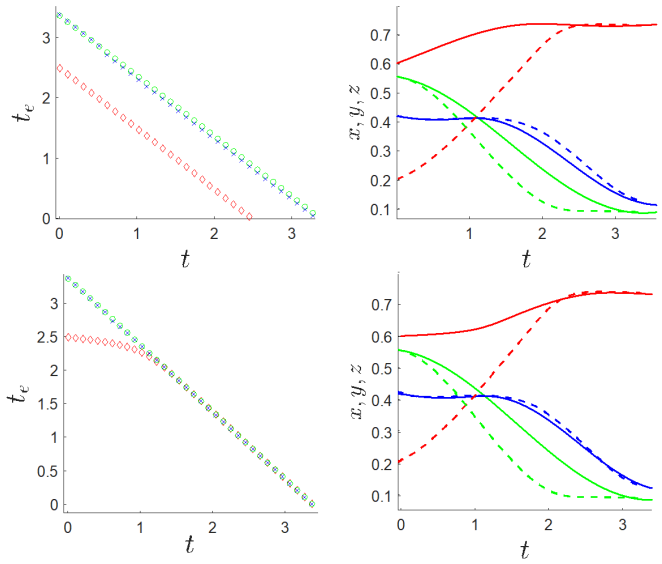

Fig. 5: Three dimensional trajectory with and without time synchronization applied in the bottom and top plots, respectively. The red, blue, and green curves are x, y, and z, respectively. The components of the demonstrated trajectory are shown in dashed lines.

model the robot's end effector orientation R as a function of any arbitrary dimension's t_e. A standard control algorithm can then be implemented to drive the error between the reference orientation $R = f(t_e)$ and the measured orientation R_m to zero.

F. Collision avoidance

One of the strengths of the PSM is the ability to transition from one phase space state to another with continuous velocity. In the presence of an obstacle, if a collision is detected, $\dot{x}_n$ can be set to zero and x_n can be set to x_{ob}, where x_{ob} is the obstacle location. To limit the maximum acceleration of the trajectory and required stopping distance, k_n is set to zero. Equation (14) is adjusted again to include the obstacle avoidance condition,

$$\ddot{x} = \begin{cases} k_n x + c_n - T\dot{x}(t_E - t_e) & \dot{x} \geq \frac{x_n - x_c}{|x_n - x_c|} \\ m & \dot{x} < \frac{x_n - x_c}{|x_n - x_c|} \\ \frac{-\dot{x}_c^2}{2(x_{ob} - x_c)} & CD \end{cases} \quad (15)$$

where CD is a collision detected flag. For our implementation, if the line segment from the current position to the goal position intersects an obstacle, then the collision condition is triggered.

IV. Experiments

We present results from a set of three experiments that highlight the different properties of the proposed PSM. We also compare the performance of PSM to DMP for a cylinder rolling task. A video of the experiments has been submitted with this paper.

A. Controller configuration

All trajectories were demonstrated to the robot via kinesthetic teaching. Joint angles were recorded at 500 Hz for each demonstration. The end effector position and orientation were calculated with forward kinematics and then smoothed with a Gaussian kernel to give a better approximation of the numerical derivatives. One PSM was used for the x, y, and z components of each demonstration. We coupled the end effector orientation with the z component's time of completion parameter t_e. For each task, the trajectory segment spatial intervals were defined with respect to the goal position.

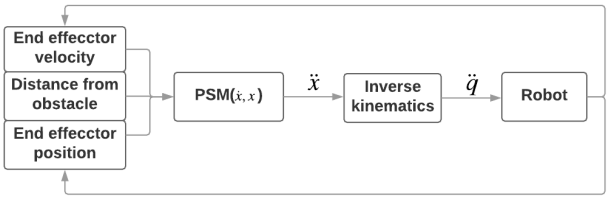

Fig. 6: Control diagram for the PSM integration with a robot

The control diagram in fig. 6 shows how the PSM was integrated with a robot. The Cartesian state of the robot's end effect was fed into the PSM, which generated Cartesian accelerations. An inverse kinematic controller then translated the PSM output to joint angle accelerations ($\ddot{q}$) and sent them to the robot.

B. Experiment 1: feeding assistant task

In this experiment, we teach the robot to perform a sequential task: feeding someone with a spoon. The task consists of five trajectories, including: grasping a spoon, bringing it to a neutral position, reaching for a bowl, performing a scooping motion, and reaching to a person for feeding. Out of the five trajectories, three require goal adaption (see fig. 7). The position of the spoon, bowl, and user were varied to evaluate the PSM's ability to adapt different goal locations. The bowl and spoon were placed at designated spots on a grid and the participant's face location was determined with a standard face recognition algorithm. Fig. 7 shows the experimental setup for this task. Four trajectories are shown in fig. 7: reaching for the spoon in blue, returning to neutral in black, reaching for the bowl in orange, and reaching for the user in red. Goal positions were evaluated on a 3x3 grid for the bowl placement, spoon placement, and user location as seen in fig. 7, resulting in 45 different trajectory adaptations.

C. Experiment 2: collision avoidance

The setup for the second experiment was the same as the feeding assistant task, except an obstacle was placed in the path when the robot brought the spoon to the user. Fig. 8 shows a snapshot of the robot motion in the presence of the obstacle. The robot was given the collision bounds of the obstacle in advance. In this scenario, the robot's y component decreased its velocity due to the predicted collision. Once the z component cleared the height of the obstacle, the y component resumed its normal trajectory toward the user.

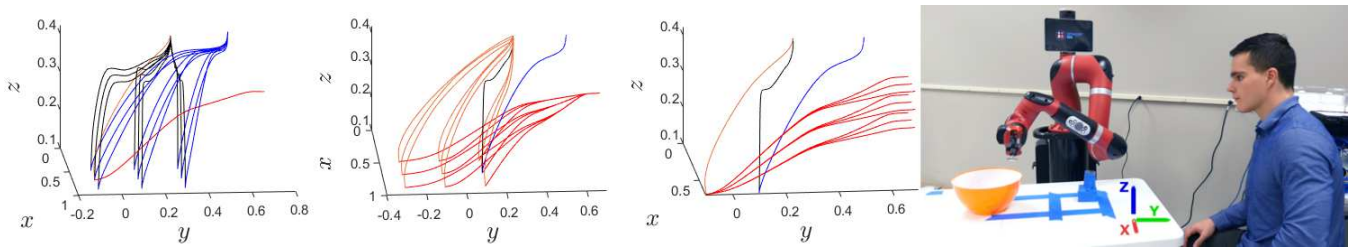

Fig. 7: Variety of goal adaptations for three sub-task of the feeding process. The experimental setup is shown on the right.

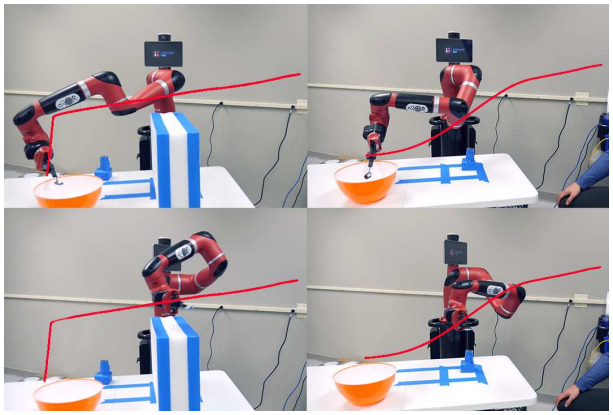

Fig. 8: The obstacle free task is shown on the right and obstacle avoidance task is shown on the left.

D. Experiment 3: cylinder rolling task

The task was to roll a cylinder from a designated start position into a basket placed at the end of the table (see fig. 9). Rolling the cylinder the correct distance required the robot's end effector to come in contact with it at a low velocity and then accelerate to a high velocity. We evaluated the PSM's phase space based goal against the DMP's time-position based goal. This task is well suited for the PSM because it requires the end effect to achieve several phase space states along the trajectory.

Fig. 9: Experimental setup for the cylinder rolling task. The cylinder is place at a designated starting point and the robot attempts to roll the cylinder into the basket at the end of the table.

Fig. 10 Shows the phase space curves for both the PSM and DMP trajectories for this task from several stating positions. The demonstration's phase space curves are shown in black. For this task, the cylinder is primary being rolled in the y direction, hence, the y component velocity is crucial in whether the cylinder will have enough momentum to roll across the table. The PSM trajectories reach a maximum

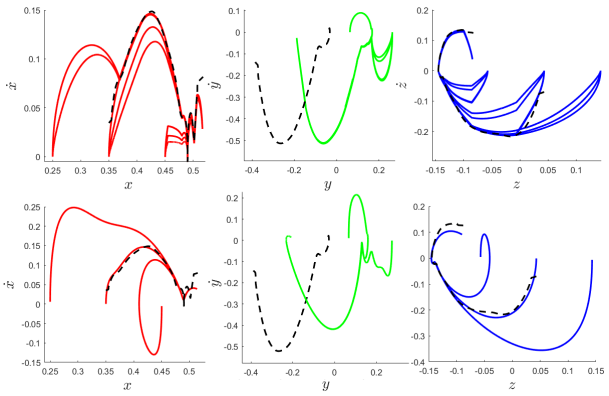

Fig. 10: The three plots on the top show the phase space curves produced by the PSM for several starting locations. The three plots on the bottom show the phase space curves for the DMP. The cylinder was place at $y = 0.1\ m$.

speed of $.5\ \frac{m}{s}$ at $y = -.1\ m$, while the DMP trajectories only reach a maximum speed of $.4\ \frac{m}{s}$. This results in task failure for this experimental configuration.

V. Conclusion

The PSM proposed in this paper originated from the concept of phase space in dynamical systems. We model trajectories by first segmenting the phase space into spatial regions, where position-acceleration relations are approximated with linear second order systems. Spatial and temporal flexibility of this approach is demonstrated through experiments with a robot. The advantage of the PSM's phase space based goal compared to DMP's time-position based goal is demonstrated in the cylinder rolling task. The benefits of this framework are robustness to temporal perturbation via time invariant dynamics, multidimensional synchronization, learning from a single demonstration, and the ability to transition between phase space states with continuous velocity via PSTFs.

Acknowledgment

This work was supported in part by the National Science Foundation (IIS 1830597).

References

[1] S. R. Ahmadzadeh, R. Kaushik, and S. Chernova. Trajectory learning from demonstration with canal surfaces: A parameter-free approach. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 544–549. IEEE, 2016.

[2] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 391–398. IEEE/ACM, 2012.

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[4] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell. Statistical dynamical systems for skills acquisition in humanoids. In *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, Osaka, Japan, 2012.

[5] S. Chernova and A. L. Thomaz. Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121, 2014.

[6] A. B. D. Pongas and S. Schaal. Rapid synchronization and accurate phase-locking of rhythmic motor primitives.

[7] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation*, 25:328373, 2013.

[8] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *In IEEE International Conference on Robotics and Automation*, page 13981403. IEEE, 2002.

[9] S. M. Khansari-Zadeh and A. Billard. Imitation learning of globally stable non-linear point-to-point robot motions using nonlinear programming. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2676–2683, Oct 2010.

[10] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, Oct 2011.

[11] S. M. Khansari-Zadeh, K. Kronander, and A. Billard. Learning to play minigolf: A dynamical system-based approach. *Advanced Robotics*, 26:1967–1993, 2012.

[12] J. Kober, K. Mlling, O. Krmer, C. H. Lampert, B. Schölkopf, and J. Peters. Movement templates for learning of hitting and batting. In *2010 IEEE International Conference on Robotics and Automation*, pages 853–858, May 2010.

[13] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47:7991, 2004.

[14] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[15] C. Paxton, F. Jonathan, M. Kobilarov, and G. D. Hager. Do what i want, not what i did: Imitation of skills by planning sequences of actions. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 3778–3785. IEEE, 2016.

[16] J. Peters and S. Schaal. Policy gradient methods for robotics. In *Int. Conf. Intelligent Robots and Systems*. IEEE, 2006.

[17] A. B. S. Calinon, F. Guenter. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(2):286–298, 2007.

[18] A. B. S. Mohammad Khansari-Zadeh. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.