

# Trajectory Representation Using Sequenced Linear Dynamical Systems

Kevin R. Dixon and Pradeep K. Khosla  
Electrical & Computer Engineering  
Carnegie Mellon University  
Pittsburgh, PA 15213  
Email: {krd, pkk}@cs.cmu.edu

**Abstract**—In this paper we present a novel approach for representing trajectories using sequenced linear dynamical systems. This method uses a closed-form least-squares procedure to fit a single Linear Dynamical System (LDS) to a simple trajectory. These LDS estimates form the elemental building blocks used to describe complicated trajectories through an automatic segmentation procedure that can represent complicated trajectories with high accuracy. Each estimated LDS induces a control law, mapping current state to desired state, that encodes the target trajectory in a generative manner. We provide a proof of stability of the control law and show how multiple trajectories can be incorporated to improve the generalization ability of the system.

## I. INTRODUCTION

One of the key issues in machine learning is feature representation. When the objective is learning from human demonstrations, then the central idea is the representation of trajectories. Learning By Observation (LBO) systems observe users performing tasks and synthesize the information to reproduce the task and achieve a goal, potentially in previously unseen conditions. These systems must be able to incorporate information from multiple demonstrations, including those occurring in different environments. In this paper we consider the representation of trajectories resulting from user demonstrations. Trajectory representation for LBO systems has somewhat different requirements than traditional robotics, since these systems must be able to reproduce tasks in previously unseen environments. In other words, LBO systems must emulate “what the user would have done” in these novel conditions. Consequently, it is essential that these systems represent trajectories in a *generative* manner, a control law mapping the current state of the system to the desired state. To facilitate learning, the representation should reduce the number of parameters needed to specify the trajectory, and similar trajectories should have similar parameters. The representation must be encoded so that it can be mapped to different environments easily to incorporate trajectories demonstrated in different environments. To address these requirements, we propose a trajectory-representation approach based on sequenced linear dynamical systems. A single Linear Dynamical System (LDS) can represent a trajectory in an extremely compact form by inducing a simple control law. Due to its simplicity, a single LDS cannot faithfully reproduce the complicated trajectories needed by many LBO systems.

However, we create sophisticated behavior from a collection of simple components by segmenting complicated trajectories into simple ones, with each segment represented by a single LDS. Complicated trajectories can then be reproduced with high accuracy using the LDS estimates in a sequential fashion.

From a high-level perspective, we first develop the concept that a single LDS can represent a simple trajectory (Section II). We use a closed-form least-squares procedure to estimate the optimal LDS for that trajectory, in Equation 3. The control law induced by the LDS is then used to reproduce the trajectory (Section III). We call the endpoint of the trajectory an *attractor* because the control law tends, or is attracted, to that point. This generative representation is desirable since, for example, modifying the initial conditions causes the LDS to adapt its response to novel situations, as in Figure 1. Similarly, we can modify the attractor to reproduce the trajectory in a different part of the workspace. By combining multiple demonstrations of a trajectory, we can produce a better generalization of “what the user would have done” in a wider variety of situations. Since the method is based on the least-squares principle, it is straightforward to incorporate multiple demonstrations into the estimated LDS, and we derive this formulation in Section IV. Because the representation is based on a control law, its stability is extremely important. In Section V, we show that under reasonable conditions the control law will produce bounded trajectories that terminate at the desired attractor point. Specifically, we provide a proof sketch that an estimated LDS is stable in the sense of Lyapunov.

To represent more complicated trajectories, we segment these trajectories into a sequence of simple trajectories, where each segment is represented by the single LDS building block. In Section VI, we derive an online method for automatically segmenting a complicated trajectory using a prediction-error criterion. We demonstrate how the sequence of LDS estimates reconstructs a complicated trajectory in Section VII. The figures in this paper use two-dimensional examples for illustrative purposes only; the ideas and the derivation of the LDS estimates generalize to arbitrary dimension, as well as state-variable derivatives such as velocity and acceleration. Likewise, the proof of stability is independent of the observation dimension.

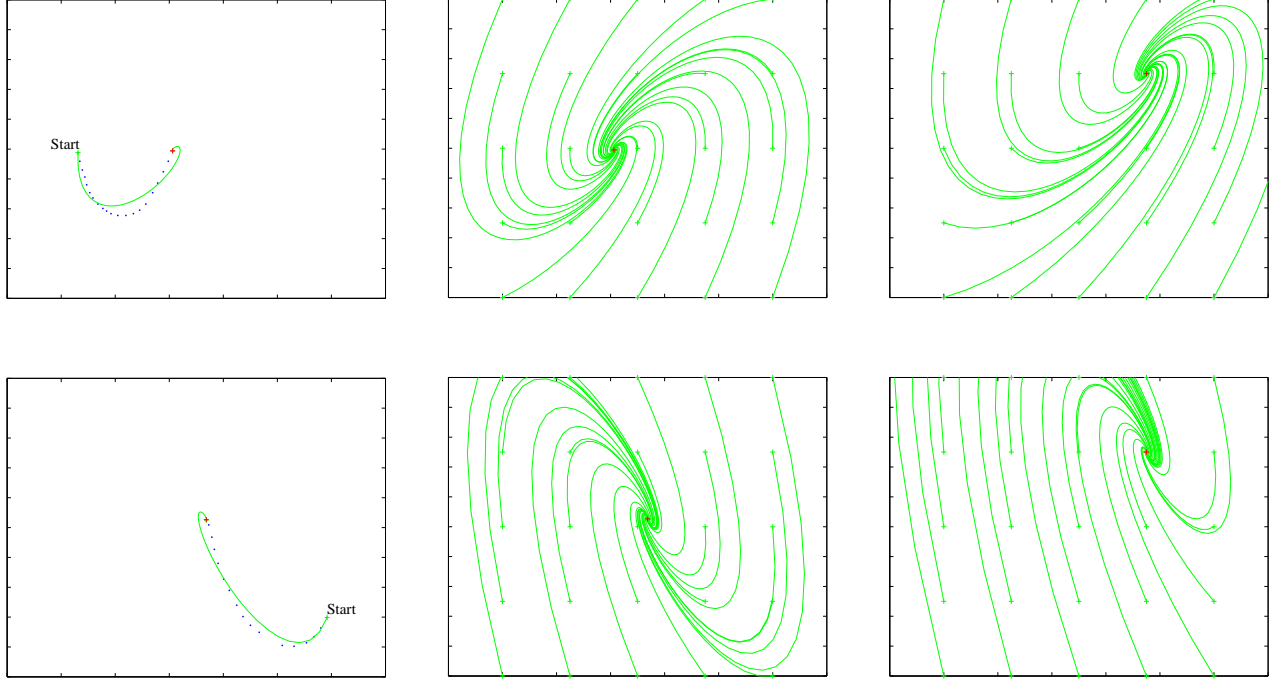


Fig. 1. The left column shows two simple trajectories fitted by an LDS according to Equation 3. The dotted line represents the samples of the user trajectory and the solid line represents the trajectory reconstructed by Equation 4. The middle column shows the response of the LDS to various initial conditions and the right column shows the response of the LDS when the attractor is shifted.

## II. SIMPLE TRAJECTORY WITH A KNOWN ATTRACTOR POINT

Consider a trajectory sampled at discrete intervals,  $\mathbf{X} = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N\}$ , where each observation is an  $(m \times 1)$  real vector and there are more samples than rows in an observation,  $N \geq m$ . Our approach attempts to find an LDS that captures the “shape” of the trajectory and terminates at the final observation. In other words, the control law induced by the LDS should be stable and have its unique attractor point at  $\mathbf{x}_N$ . A discrete-time LDS with constant input is given by the difference equation

$$\mathbf{x}_{n+1} = \mathbf{A}\mathbf{x}_n + \mathbf{B}u.$$

Assuming the system is stable and the matrix  $(\mathbf{I} - \mathbf{A})^{-1}$  exists, then the unique attractor,  $\mathbf{x}_\infty$ , is computed as

$$\mathbf{x}_\infty = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}u.$$

Consequently, if we choose  $\mathbf{B} = (\mathbf{I} - \mathbf{A})$ , then the input is the point of attraction,  $\mathbf{x}_\infty = u$ . Using the LDS formulation, we assume that the observations are generated according to

$$\begin{aligned} \mathbf{x}_{n+1} &= (\mathbf{I} + \mathbf{R})\mathbf{x}_n - \mathbf{R}\mathbf{x}_N \\ &= \mathbf{R}(\mathbf{x}_n - \mathbf{x}_N) + \mathbf{x}_n. \end{aligned} \quad (1)$$

We give Equation 1 this particular form since if the matrix  $(\mathbf{I} + \mathbf{R})$  is stable, then the attractor of Equation 1 is  $\mathbf{x}_N$ , since  $\mathbf{B} = (\mathbf{I} - \mathbf{A})$ . Given a trajectory, we rewrite Equation 1

as the column-stacked matrix equation

$$\begin{aligned} [\mathbf{x}_1 \cdots \mathbf{x}_N] &= \mathbf{R}([\mathbf{x}_0 \cdots \mathbf{x}_{N-1}] - [\mathbf{x}_N \cdots \mathbf{x}_N]) \\ &\quad + [\mathbf{x}_0 \cdots \mathbf{x}_{N-1}], \\ &\Rightarrow \\ \mathbf{X}_{1:N} &\doteq \mathbf{R}(\mathbf{X}_{0:N-1} - \mathbf{\Gamma}_N) + \mathbf{X}_{0:N-1}. \end{aligned} \quad (2)$$

The least-squares solution for  $\mathbf{R}$  is

$$\hat{\mathbf{R}} = (\mathbf{X}_{1:N} - \mathbf{X}_{0:N-1})(\mathbf{X}_{0:N-1} - \mathbf{\Gamma}_N)^R, \quad (3)$$

where  $\mathbf{C}^R$  denotes the right pseudoinverse of  $\mathbf{C}$ . If the matrix  $\mathbf{C}$  has full row rank, then  $\mathbf{C}^R = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}$ . The matrix  $\hat{\mathbf{R}}$  captures the salient features of the trajectory such as direction, curvature, and speed of the trajectory. The attractor,  $\mathbf{x}_N$ , specifies the trajectory terminus.

## III. REPRODUCING A SIMPLE TRAJECTORY

To reproduce the trajectory, the LDS is initialized with  $\hat{\mathbf{x}}_0 \equiv \mathbf{x}_0$ , which induces from Equation 1 the autonomous control law

$$\hat{\mathbf{x}}_{n+1} = \hat{\mathbf{R}}(\hat{\mathbf{x}}_n - \mathbf{x}_N) + \hat{\mathbf{x}}_n. \quad (4)$$

This process repeats until reaching the stopping criterion,  $\|\hat{\mathbf{x}}_n - \mathbf{x}_N\| \leq \tau$ , where  $\tau > 0$  is some scalar threshold. Since the trajectory is reproduced using an autonomous control law, it can be modified by shifting the initial conditions, the attractor point, or both. There are no complicated scaling laws that plague other trajectory-representation methods, such as

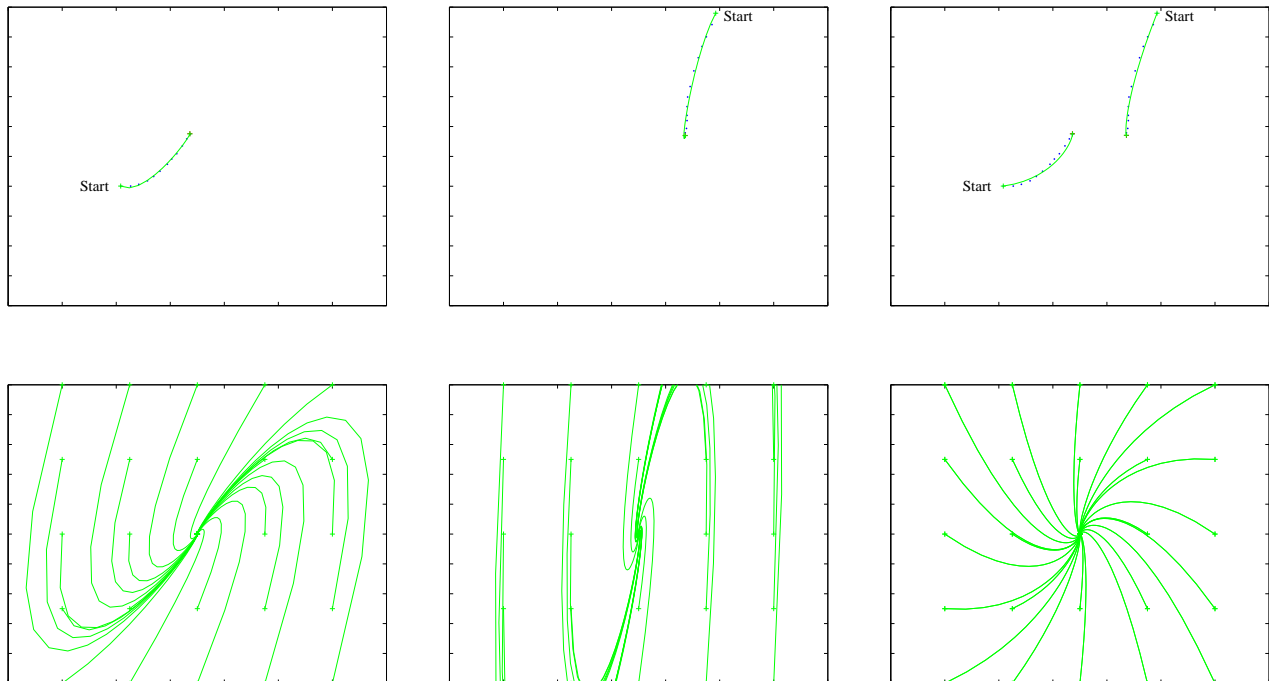


Fig. 2. The left and middle columns show two examples of trajectories with a slight counter-clockwise curvature, with the original trajectories on top and the response of the LDS on bottom. Individually, neither estimate produces the intended generalization. In the right column is the combined estimate of the two trajectories, computed from Equation 5, which yields the desired slight counter-clockwise generalization.

spline interpolation [1]. Modifying either the initial conditions or attractor will cause the estimated trajectory to stretch, shrink, etc. Shifting the initial conditions and the attractor by the same amount would cause Equation 4 to reproduce a trajectory of the same shape, simply offset by the shift amount. In Figure 1, we use Equation 3 to fit an LDS through two trajectories and Equation 4 to reproduce the estimated trajectories. We also show the response of the LDS to various initial conditions and its response to a shifted attractor point. The response to the different initial conditions corresponds to the hypothesis of “what the user would have done” in that situation.

When applied to the real-time control of a system, such as a robot, then the deterministic iteration in Equation 4 then becomes the desired state of the system. Like other researchers [2], we assume that the system can observe its current state,  $\hat{\mathbf{x}}_n$ , and that there exists a low-level controller that is able to actuate toward a desired state,  $\hat{\mathbf{x}}_{n+1}$ . However, if the system is nonholonomic or underactuated, then it may not be able to achieve the desired state within the allotted time. At each iteration of the control locus, the system senses its new state and computes a new desired state. In this sense, Equation 4 computes the sequence of states specifying *how* to achieve a goal even if the system diverges from the intended trajectory.

#### IV. COMBINING MULTIPLE TRAJECTORIES

Since our representation method uses a least-squares procedure, it is simple to incorporate multiple examples of a trajec-

tory to create an estimate that generalizes better. Suppose we have two demonstrations of a trajectory,  $\mathbf{X}^1 = \{\mathbf{x}_0^1, \dots, \mathbf{x}_{N_1}^1\}$  and  $\mathbf{X}^2 = \{\mathbf{x}_0^2, \dots, \mathbf{x}_{N_2}^2\}$ , though in principle and in practice the following method works with an arbitrary number of demonstrations. We rewrite Equation 2 by column-stacking both trajectories and define the matrices

$$\begin{aligned} \mathbf{X}_{1:N}^{1,2} &\triangleq [\mathbf{x}_1^1 \cdots \mathbf{x}_{N_1}^1 \mathbf{x}_1^2 \cdots \mathbf{x}_{N_2}^2]; \\ \mathbf{X}_{0:N-1}^{1,2} &\triangleq [\mathbf{x}_0^1 \cdots \mathbf{x}_{N_1-1}^1 \mathbf{x}_0^2 \cdots \mathbf{x}_{N_2-1}^2]; \\ \mathbf{\Gamma}_N^{1,2} &\triangleq [\mathbf{x}_{N_1}^1 \cdots \mathbf{x}_{N_1}^1 \mathbf{x}_{N_2}^2 \cdots \mathbf{x}_{N_2}^2]. \end{aligned}$$

Similar to Equation 3, the least-squares estimate for these combined trajectories is

$$\hat{\mathbf{R}}^{1,2} = \left( \mathbf{X}_{1:N}^{1,2} - \mathbf{X}_{0:N-1}^{1,2} \right) \left( \mathbf{X}_{0:N-1}^{1,2} - \mathbf{\Gamma}_N^{1,2} \right)^R. \quad (5)$$

In Figure 2, we provide two examples of a simple trajectory, both with a slight counter-clockwise curvature. The LDS fitted to each individual trajectory is optimal in the least-squares sense. Considering the response of these control laws to various initial conditions, it is clear that the individual estimates do not generalize to “slight counter-clockwise curvature.” However, computing the least-squares estimate of the *combined* trajectories produces the intended generalization.

#### V. STABILITY OF THE LDS ESTIMATE

Define the vector  $\delta_n \triangleq \mathbf{x}_n - \mathbf{x}_N$  and the matrix

$$\begin{aligned} \Delta_{1:N} &\triangleq \mathbf{X}_{1:N} - \mathbf{\Gamma}_N \\ &\equiv [\delta_1 \cdots \delta_N]. \end{aligned}$$

The matrix  $\Delta_{0:N-1}$  is defined similarly. If the matrix  $\Delta_{0:N-1}$  has full row rank, then we can rewrite Equation 3 as

$$\begin{aligned}\widehat{\mathbf{R}} &= (\mathbf{X}_{1:N} - \mathbf{X}_{0:N-1})(\mathbf{X}_{0:N-1} - \Gamma_N)^R \\ &= ((\mathbf{X}_{1:N} - \Gamma_N) - (\mathbf{X}_{0:N-1} - \Gamma_N))(\mathbf{X}_{0:N-1} - \Gamma_N)^R \\ &= (\mathbf{X}_{1:N} - \Gamma_N)(\mathbf{X}_{0:N-1} - \Gamma_N)^R - \mathbf{I} \\ &\doteq \Delta_{1:N}\Delta_{0:N-1}^R - \mathbf{I}.\end{aligned}$$

We note that the estimated LDS is

$$\begin{aligned}\mathbf{x}_{n+1} &= \widehat{\mathbf{R}}(\mathbf{x}_n - \mathbf{x}_N) + \mathbf{x}_n \\ &= (\widehat{\mathbf{R}} + \mathbf{I})\mathbf{x}_n - \widehat{\mathbf{R}}\mathbf{x}_N \\ &\doteq (\Delta_{1:N}\Delta_{0:N-1}^R)\mathbf{x}_n - \widehat{\mathbf{R}}\mathbf{x}_N.\end{aligned}$$

There are two broad approaches to ensure the stability of a discrete time-invariant dynamical system. The primary method for *linear* systems is showing that the system matrix is *convergent*, i.e., having eigenvalues inside the unit circle [3]. The magnitude of the largest eigenvalue of a matrix  $\mathbf{A}$  is called the *spectral radius* and is written  $\rho(\mathbf{A})$ . If the matrix  $\mathbf{A}$  is constant, then the eigenvalues can be evaluated directly to ensure stability. When the system matrix is determined algorithmically from inputs unknown *a priori*, then properties of the spectral radius must be invoked. For instance, it is well known that  $\rho(\mathbf{A}) \leq \|\mathbf{A}\|_p$ , where  $\|\cdot\|_p$  is any of the matrix  $p$ -norms [4]. We note that the LDS estimate is computed from the product of two matrices,  $\Delta_{1:N}\Delta_{0:N-1}^R$ . There are recent results indicating that bounding the eigenvalues of a product of two matrices is, in general, *undecidable* [5]. Even with this discouraging result, it is certainly possible to place restrictions on the constituent matrices to bound the eigenvalues of the product. Using this approach, we investigated a variety of conditions to guarantee stability by restricting the class of acceptable trajectories. Unfortunately, we were unsuccessful in deriving any *useful, reasonable, or meaningful* conditions. Our difficulty seems in line with anecdotal evidence from other researchers [6].

The more general approach for ensuring stability of a linear or nonlinear dynamical system is by Lyapunov's direct (or second) method [3]. In the DT LDS case, for stability in the sense of Lyapunov (i.s.L.), it is sufficient to find a symmetric positive definite (PD) matrix  $\tilde{\mathbf{P}}$  such that  $\tilde{\mathbf{P}} - \mathbf{A}^T\tilde{\mathbf{P}}\mathbf{A} = \tilde{\mathbf{Q}}$ , where  $\tilde{\mathbf{Q}}$  is some symmetric positive semidefinite (PSD) matrix [3]. Since the eigenvalues of  $\mathbf{A}$  and  $\mathbf{A}^T$  are the same, stability i.s.L. can be determined equivalently by choosing a symmetric PD matrix  $\mathbf{P}$  such that

$$\mathbf{P} - \mathbf{A}\mathbf{P}\mathbf{A}^T = \mathbf{Q}, \quad (6)$$

where  $\mathbf{Q}$  is some symmetric PSD matrix. This equation sometimes goes by the name of the Discrete Algebraic Lyapunov Equation (DALE). Finding the matrix  $\mathbf{P}$  is more of an art than a science and typically relies on problem-specific intuition. Applying this intuition sidesteps the undecidability problem of bounding the eigenvalues of a product of arbitrary matrices mentioned earlier.

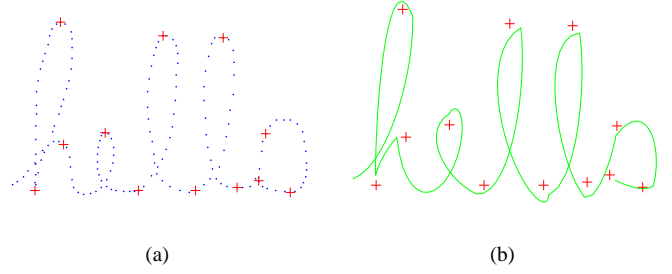


Fig. 3. Trajectory of the cursive word ‘hello’ with attractors extracted automatically using Equation 7 (Figure 3(a)). The green curve indicates the estimated trajectory from Equation 4 (Figure 3(b)).

*Theorem 1:* Equation 4 is stable i.s.L. about the attractor  $\mathbf{x}_N$  if the matrix  $\Delta_{0:N-1}$  has full row rank.

*Proof sketch:* We evaluate the DALE, Equation 6, with  $\mathbf{P} = \Delta_{0:N-1}\Delta_{0:N-1}^T$ . After some intricate algebra, the DALE then evaluates to

$$\mathbf{Q} = \delta_0\delta_0^T + \Delta_{1:N} \left( \sum_{i=1}^m \mathbf{v}_i\mathbf{v}_i^T \right) \Delta_{1:N}^T,$$

where  $\mathbf{v}_i$  is the  $i$ th column of the matrix  $\mathbf{V}$  resulting from the singular value decomposition of  $\Delta_{0:N-1}$ . Clearly the matrix  $\mathbf{Q}$  is symmetric PSD and, consequently, Equation 4 is stable i.s.L. about the attractor  $\mathbf{x}_N$ . ■

The assumptions of Theorem 1 imply that if the trajectory is anything other than a perfect line, then Equation 3 will produce bounded trajectories that terminate at the desired attractor. However, as observations become increasingly collinear there could be numerical-precision problems, causing the matrix  $\Delta_{0:N-1}$  to become effectively rank deficient. This is typically manifested by the estimated matrix  $\widehat{\mathbf{R}}$  becoming poorly conditioned, and can be determined by checking if the condition number exceeds some threshold,  $\kappa(\widehat{\mathbf{R}}) > \kappa_{\max}$ . A poorly conditioned matrix is then replaced by a scaled identity matrix. Since every vector is an eigenvector of a scaled identity matrix, Equation 4 will then induce a control law that produces straight lines terminating at the attractor from any initial condition.

## VI. COMPLICATED TRAJECTORIES WITH UNKNOWN ATTRACTOR POINTS

In this section, we consider trajectories that cannot be faithfully reproduced using a single LDS estimate. We formulate the problem as finding a *sequence* of LDS estimates that represents the complicated trajectory. Furthermore, we assume that there are no auxiliary signals indicating an appropriate segmentation. Somewhat arbitrarily, we pursued a method that allows the system to perform the segmentation *online*, as observations become available, instead of a batch-processing method that requires the trajectory be complete. It appears that finding the optimal segmentation of a trajectory has no closed-form solution and is heuristic in nature, similar to determining the optimal number of clusters in the  $k$ -means algorithm. Our segmentation heuristic is based on the predictability of

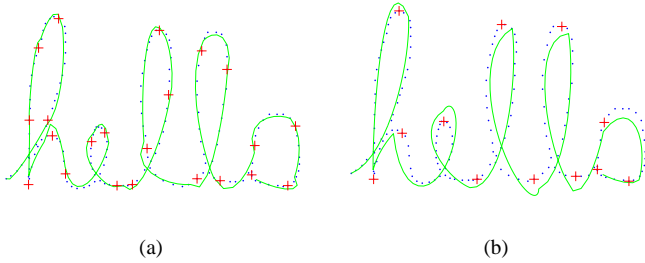


Fig. 4. Representation of the cursive word “hello” from different values of the prediction-error threshold. The segmentation with  $\varepsilon_{\max} = 0.4$  in Figure 4(a) uses 22 matrix-attractor pairs. The segmentation with  $\varepsilon_{\max} = 0.8$  in Figure 4(b) uses 12 matrix-attractor pairs.

subsequences of the trajectory, where its observations can be modeled by a single LDS. At an unknown point in time,  $N_i$ , the trajectory is no longer well represented by the LDS that generated recent observations. To determine this segmentation, at each time step we compute a prediction of the next observation in the trajectory. If the prediction is sufficiently accurate, then we consider the current LDS estimate appropriate. If the prediction is inaccurate, then a new LDS should be used. Specifically, at time  $n$  we estimate an LDS according to Equation 3 between time  $N_{i-1}$  and time  $n-1$  and write this matrix as  $\hat{\mathbf{R}}_{N_{i-1}:n-1}$ . We assign  $\mathbf{x}_n$  as the attractor of the LDS and predict the previous observation,

$$\hat{\mathbf{x}}_{n-1} = \hat{\mathbf{R}}_{N_{i-1}:n-1} (\mathbf{x}_{n-2} - \hat{\mathbf{x}}_n) + \mathbf{x}_{n-2}.$$

We compute prediction error as

$$\varepsilon_n \triangleq \frac{\|\mathbf{x}_{n-1} - \hat{\mathbf{x}}_{n-1}\|}{\|\mathbf{x}_{n-1} - \mathbf{x}_{n-2}\|}. \quad (7)$$

This definition attempts to normalize the error so that its value is independent of the speed or sampling rate of the trajectory. For instance, if the user moves very quickly, then we expect predictions to be less accurate on an absolute scale. On the other hand, when a trajectory is sampled at a relatively high rate, we expect predictions to be more accurate on an absolute scale, since not much time has elapsed between observations. Normalizing by the distance between observations accounts for this discrepancy, so that an inaccurate prediction is somewhat invariant with respect to these issues. If the prediction error exceeds a predetermined threshold,  $\varepsilon_n > \varepsilon_{\max}$ , then this implies that the observation  $\mathbf{x}_n$  is not an appropriate attractor for this segment. Therefore, we assign the segmentation time as  $N_i = n-1$  and the attractor as  $\mathbf{x}_{N_i} = \mathbf{x}_{n-1}$ . We then segment the trajectory between time  $N_{i-1}$  and time  $N_i$  and compute the corresponding LDS as in Equation 3,

$$\hat{\mathbf{R}}_{N_i} = (\mathbf{X}_{N_{i-1}+1:N_i} - \mathbf{X}_{N_{i-1}:N_{i-1}}) (\mathbf{X}_{N_{i-1}:N_{i-1}} - \mathbf{\Gamma}_{N_i})^R.$$

After the entire trajectory has been segmented, a sequence of  $M$  LDS estimates,  $\{(\hat{\mathbf{R}}_{N_1}, \mathbf{x}_{N_1}), \dots, (\hat{\mathbf{R}}_{N_M}, \mathbf{x}_{N_M})\}$ , now describes the complicated trajectory. In most cases this sequence of matrix-attractor pairs will reduce the number of

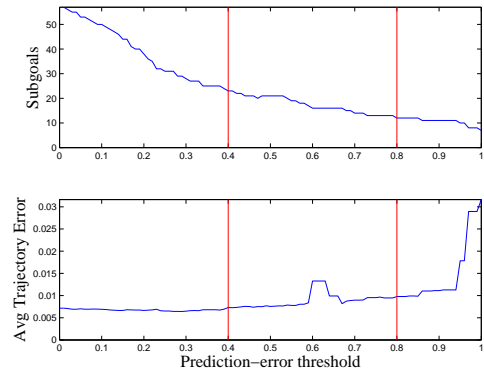


Fig. 5. Trade-off between simplicity and accuracy: number of matrix-attractor pairs and average error as a function of prediction-error threshold  $\varepsilon_{\max}$ .

parameters needed to represent the trajectory. Let  $N$  be the number of  $(m \times 1)$  observations in the original trajectory. The number of parameters used to specify the trajectory is then  $Nm$ . With the sequenced LDS method the number of parameters needed to specify the approximated trajectory is  $M(m^2 + m)$ , where  $M$  is the number of LDS estimates, since  $m^2$  numbers are needed for the matrix  $\hat{\mathbf{R}}_{N_i}$  and  $m$  for the attractor  $\mathbf{x}_{N_i}$ . In Figure 3, we apply the segmentation algorithm to the trajectory of the cursive word “hello.” The original trajectory is specified by 350 numbers. The LDS representation used 12 matrix-attractor pairs, or 72 numbers, a reduction of almost 80%. However, the reduced-parameter representation results in an error between the original and estimated trajectories. Like many algorithms, there is a tradeoff between the expressiveness of the system and the number of parameters used. In Figure 4, we segment the cursive word “hello” with high and low values of the prediction-error threshold,  $\varepsilon_{\max}$ . Qualitatively, the estimated trajectory in Figure 4(b) is less faithful to the original trajectory than the estimate in Figure 4(a), which uses nearly twice as many matrix-attractor pairs.

To quantify this trade-off between simplicity and accuracy, we compute the error between the original and estimated trajectories. The error is defined to be the norm between an observation in the original trajectory and the closest point in the estimated trajectory. In Figure 5, we plot the resulting average error and number of matrix-attractor pairs as a function of the prediction-error threshold,  $\varepsilon_{\max}$ . The curves show the anticipated result: as the number of parameters decreases, the average error generally increases. The *optimal* value for the prediction-error threshold is task dependent: some tasks require high accuracy, while others favor a simpler representation. As such, the correct value of  $\varepsilon_{\max}$  depends on the task at hand.

## VII. REPRODUCING A COMPLICATED TRAJECTORY

To reproduce a complicated trajectory, given a sequence of matrix-attractor pairs, we simply initialize the difference equation with  $\hat{\mathbf{x}}_0 = \mathbf{x}_0$  and repeatedly apply

$$\hat{\mathbf{x}}_{n+1} = \hat{\mathbf{R}}_{N_1} (\hat{\mathbf{x}}_n - \mathbf{x}_{N_1}) + \hat{\mathbf{x}}_n,$$

until achieving  $\|\hat{x}_n - x_{N_1}\| \leq \tau$ . At this point, the next matrix-attractor pair is used ( $\hat{R}_{N_2}, x_{N_2}$ ), and so forth, until the final attractor point is reached, cf. Figure 3(b). Since each LDS estimate is stable i.s.L., any trajectories generated by the difference equations will be bounded and terminate at the final attractor point in finite time. Finite termination is guaranteed regardless of initial conditions *or* deviations from the prescribed control law.

### VIII. RELATED WORK

The trajectory-representation method presented in this paper forms the substrate for the mobile-robot LBO system described by Dixon and Khosla [7]. The LBO system uses the sequenced LDS representation to describe the trajectory of a user demonstrating a task, sensed from a scanning laser range finder. The attractors extracted from the trajectory are associated with objects in the environment and, as these objects move, the attractors move accordingly. The control law from the LDS representation interpolates the trajectory with the modified attractor. This endows the LBO system with the ability to learn from, and perform, demonstrations in different environments.

According to Ijspeert et al. [8], “there seems to be consensus that among the most important desirable properties of movement encoding are: 1) the ease of representing and learning a goal trajectory, 2) compactness of the representation, 3) robustness against perturbations and changes in a dynamic environment, 4) ease of re-use for related tasks and easy modification for new tasks, and 5) ease of categorization for movement recognition.” To varying degrees, these criteria are applicable to the class of systems that we consider. We would also recommend the ability to incorporate multiple examples of a trajectory in order to capture the *intentions* of the user better. The most obvious method for representing a trajectory, cubic spline interpolation [1], is extremely sensitive to changes in initial conditions and is difficult to recompute for different operating conditions. A similar method for representing the trajectories of mobile robots using piecewise-cubic Bézier curves has also been proposed [9]. As Schaal et al. [2] note, spline methods “are not very robust in coping with unforeseen perturbations of the movement” and require “more complex computations in terms of scaling laws.” This observation underscores the need to represent trajectories in a generative manner. Other trajectory-representation methods include symbolic if-then coding [10], neural networks [11], and optimization methods [12]. Ijspeert et al. [8] describe a trajectory-representation method based on nonlinear attractor dynamics. This approach is particularly appealing since it is a generative method, providing a control law so that a system can reproduce trajectories in different environment configurations by adjusting the attractor points. This method also allows the representation of a wide variety of trajectories, including discrete and rhythmic movements. However, this formulation is sensitive to the speed with which trajectories are demonstrated, which may be undesirable in some applications. Also, entire trajectories are encoded with a single attractor and it may be difficult to modify portions of a trajectory without

complete retraining. Estimating the parameters of a linear, time-invariant dynamical system is squarely within the domain of System Identification [13]. However, this field is primarily concerned with experimental setup and unbiasedness, less with stability and compactness. In some sense, we have co-opted an elementary concept from system identification and are applying the ideas to trajectory representation.

### IX. CONCLUSIONS

In this paper, we presented a novel method for representing trajectories using sequenced linear dynamical systems. Simple trajectories are represented by a single LDS estimate while complicated trajectories use the sequenced representation. The LDS estimates are computed in closed form by a least-squares procedure. We showed that multiple demonstrations can be combined in order to improve generalization and provided a proof sketch guaranteeing stability of the LDS estimates.

### ACKNOWLEDGMENT

We would like to thank Bruce Krogh and John Dolan for very helpful discussion. We are grateful to the Intel Corporation for providing the computing hardware. This work was sponsored by ABB Corporate Research.

### REFERENCES

- [1] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 2nd ed. Addison Wesley, 1989.
- [2] S. Schaal, A. Ijspeert, and A. Billard, “Computational approaches to motor learning by imitation,” *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, vol. 358, pp. 537–547, 2003.
- [3] P. J. Antsaklis and A. N. Michel, *Linear Systems*. McGraw-Hill, 1997.
- [4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.
- [5] V. D. Blondel and J. N. Tsitsiklis, “The boundedness of all products of a pair of matrices is undecidable,” *Systems and Control Letters*, vol. 41, 2000.
- [6] V. D. Blondel, J. Theys, and A. A. Vladimirov, “An elementary counterexample to the finiteness conjecture,” *SIAM Journal on Matrix Analysis*, vol. 24, no. 4, 2003.
- [7] K. R. Dixon and P. K. Khosla, “Learning by observation with mobile robots: A computational approach,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004.
- [8] A. J. Ijspeert, J. Nakanishi, and S. Schaal, “Trajectory formation for imitation with nonlinear dynamical systems,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.
- [9] J.-H. Hwang, R. C. Arkin, and D.-S. Kwon, “Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.
- [10] M. N. Nicolescu, “A framework for learning from demonstration, generalization and practice in human-robot domains,” Ph.D. dissertation, Department of Computer Science, University of Southern California, 2003.
- [11] H. Friedrich, S. Münch, R. Dillmann, S. Bocionek, and M. Sassin, “Robot programming by demonstration (RPD): Supporting the induction by human interaction,” *Machine Learning*, vol. 23, pp. 163–189, 1996.
- [12] S. Schaal, “Learning from demonstration,” in *Advances in Neural Information Processing Systems*, M. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9, 1997, pp. 1040–1046.
- [13] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Prentice Hall, 1999.