Response to Reviewers: Manuscript ID 17-1142.1: "Learning Augmented Joint-Space Task-Oriented Dynamical Systems: A Linear Parameter Varying and Synergetic Control Approach"

March 16, 2018

First of all, we would like to thank all reviewers for the time they spent reviewing our paper. We greatly appreciate the comments, suggestions and opinions that we received. We have addressed all the necessary comments and suggestions in the attached manuscript accordingly.

Comments from Reviewer #1 (Reviewer: 4 (Review30069)):

1. *"As mentioned by the authors, there are many reasons to plan in joint space rather than task space. One of the important reasons to me seems to be able to plan end-effector orientation, not just position. Traditional movement primitives, for example DMPs, do not explicitly encode planning in the SO(3) space of orientations. There is research that addresses this question, for example [1], but they lose the simplistic appeal of movement primitives. I think it would be a great addition to this paper if the authors illustrate if their joint space DS can be used to accomplish orientation sensitive tasks, and generalize to end-effector orientations different that what was demonstrated.[1] Ude, Ale, et al. "Orientation in cartesian space dynamic movement primitives." Robotics and Automation (ICRA), 2014 IEEE International Conference on."*

   `Response:` We highly appreciate this comment. When we initially proposed this approach, we did not think about tackling the problem of reaching a target in orientation as well. Since our DS is learned in joint-space, indeed, we do not have to deal with the challenges that arise when trying to reach a target in both position and orientation as we do not have the constrains imposed by SO(3). There was theoretically no modification needed in our controller:

$$\underbrace{\dot{q}}_{\mathbb{R}^{m\times 1}} = -\underbrace{\mathcal{A}(q)}_{\mathbb{R}^{m\times m}}\underbrace{J^T(q)}_{\mathbb{R}^{m\times d}}\underbrace{(H(q)-x^*)}_{\mathbb{R}^{d\times 1}} \tag{1}$$

   Originally, when working solely in position $d = 3$, now we represent the target with $d = 9$, where the first 3 dimensions correspond to Cartesian position and the remaining 6 correspond to the first and second columns of a rotation matrix $R \in SO(3)$. This only modifies the Jacobian $J(q)$ which is now $J(q) \in \mathbb{R}^{9\times 7}$ rather than $J(q) \in \mathbb{R}^{3\times 7}$. The role of $\mathcal{A}(q)$ remains the same. The following paragraph has been incorporated, as well as some additions in the abstract and problem statement: "Finally, by learning a behavior in joint-space we can inherently reach a task-space target, not only in the Cartesian 3-D space $\mathbb{R}^3$, but also in the space of orientations $SO(3)$. Planning or learning the rotational component of motion is a challenging problem. Representing an orientation as a vector in Euclidean space may lead to inaccurate and unstable motions, due to its directional nature and vulnerability to singularities. Several works have proposed tailor-made learning approaches that consider the non-Euclidean geometry of the $SO(3)$ space to generate rotational motion [**?**] [**?**] [**?**]. These approaches, however, require an explicit coupling between position and orientation, that might cause discontinuities in the resulting end-effector motion. Such coupling is not necessary if the motion is encoded in joint-space."

   Furthermore, we have re-learnt all demonstrated behaviors incorporating orientation in the task-space target x^*. Table I has been updated accordingly, showing no deterioration in generalization and performance. Finally, as will be discussed in the response to the Comments 10 and 12, we compare our approach to a Task-Space DS learnt in both position+orientation, showing a considerable improvement in accuracy while reaching the target.

2. *"In the introduction, the authors stress that jacobian psuedo inverse can be computationally expensive. Is this really true for a 7-dof robot arm? Can the authors provide some numbers where this would really be an issue for a robot controlled at 1000Hz? I find the argument that jacobian inverse does not work for singularities and needs a set of heuristics to be more compelling. Maybe the authors should rewrite the introduction to stress more on this issue?"*

   `Response:` For a 7-DOF robot it is indeed negligible. Computational burden only becomes a real issue for higher

degrees of freedom, like a multi-fingered hand or multi-arm robotic setting. We removed this remark in order to allow space to discuss about other issues.

3. *"In Section II, criterion I, the authors talk about a positive symmetric matrix P. I do not see it in the equations above. Is this a weighing matrix?"*
   `Response:` Removed, that was a carry-over from a previous formulation.

4. *"In criterion II, the error is defined with respect to the velocity error, rather than a position or a combination of both. This means that the motion can be different from the demonstrated motion. I understand that this is important to keep the later optimization convex, but is there any disadvantage to using velocity error rather than position error?"*
   `Response:` Optimizing for position distance alone would have the effect of learning a motion that converged to the average of previous motions, thus not changing the motion in new situations or fitting the joint velocity profile that is often a desired component of the motion. Using velocity error does mean that, for motions strongly dependent on joint position rather than joint velocity, the generated motions may generalize from the demonstrations in unexpected ways. However, as you mentioned, minimizing position error would, depending on the formulation, be prohibitively difficult. This would imply simulating the learned motion forward in time to recompute the robot's positions, making the optimization much more complicated; this way, we can treat each point in the motion independently.

5. *'I do not completely understand the significance of A(q). From Figure 3, it seems like A(q) helps resolve redundancy, while achieving the task space goal. Do the authors have a mathematical (in constrast to intuitive) explanation for this? Does A(q) somehow relate to the null-space of the jacobian?"*
   `Response:` A(q) transforms a joint velocity vector approximately aligned with the task-space error into a different joint velocity vector that remains approximately aligned with the error, but which we want to choose to more-closely approximate our demonstrations. In this sense, it is resolving redundancy between all the possible joint velocity vectors that would move the controller closer to the origin. The paragraph in Section III has been modified to improve clarity. A(q) doesn't relate to the null-space of the Jacobian in any clear way, as its value can and does alter the task-space trajectory as well as the joint space trajectory. The only property that A(q) preserves is the joint velocity's moving the task space position closer to the goal, but the direction of motion is not necessarily the same as the direction of task-space error.

6. *"In Section III, ?One can intuitively understand the control law as follows:$(H(q) - x^*)$ denotes the position error w.r.t. the task target and by multiplying that error by the transposed Jacobian JT (q), the error is projected into joint space (similar to Jacobian transpose control)?. I find this explanation misleading as jacobian multiplied to position error does not project the error into joint space. Jacobian transforms velocities, not positions. Maybe the law can be seen more as a first order approximation of the error in task space. In any case, I would either clarify this statement more or take it out of the manuscript all together. It seems like the DS is just a PD law that provably converges to a fixed attractor in task space."*
   `Response:` The passage has been rewritten to remove the word "project", which you appropriately noted was incorrect, and further clarifies the nature of the DS as a PD controller. We have added the following text in the manuscript: *"One can intuitively understand the control law as follows: $(H(q) - x^*)$ denotes the position error w.r.t. the task target, which we then reinterpret as the task-space velocity of a proportional controller. By multiplying that error by the transposed Jacobian $J^T(q)$, it is transformed into a joint-space velocity vector correlated with the error (similar to Jacobian transpose control [?]), see Fig. [?]. The positive definite matrix $\mathcal{A}(q)$ warps the resulting joint-space velocity; Fig. [?] illustrates the effects of $\mathcal{A}(q)$ on the generated motion. Thus the controller can be thought of as a proportional controller in joint space.*

7. *"About figure 2 again, the position error in task and joint space are not simply multiplied by a Jacobian. Although, task space error can be first-order approximated using a Jacobian multiplication with joint space error. What does the right figure in Fig 2 exactly represent?"*
   `Response:` The figure on the left is simply the cartesian error in task space. The figure on the left is the Lyapunov potential function on which the jacobian transpose controller descends. In showing the attractive regions in joint-space corresponding to this Lyapunov function, we aim to offer insight into how the controller reformulates the task-space potential into a different yet related joint-space potential. The caption has been rewritten to add clarity.

8. *"In Section IV, the authors say that they use EM to determine the parameters of the GMM in Eq 6. Can they elaborate a little on this? What is the expectation taken over and what likelihood is maximized?"*

`Response:` The algorithm computes the expected probability of generating each of the joint positions given the current values of the GMM's means, variance matrices, and scaling values. The algorithm then chooses GMM parameters to maximize the posterior likelihood of the model, using previously-computed expected component membership as likelihood weights when computing the new mixture. Specifically, we used MATLAB's *fitgmdist* implementation. Added a citation to clarify.

9. *"The learned dimensionality reduction and parameters seem task dependant. I wonder if the authors have any idea on how to generalize this across tasks."*
`Response:` The dimensionality reduction could be generalized across tasks that had the same motor synergies - that is, for example, all tasks in which the shoulder and elbow always move together. Similarly, one could imagine several similar motions that could be decomposed into shared submotions, thus allowing the parameters to be shared. However, in general, fixing some of the parameters while optimizing will simply lead to a less-flexible model, and considering that there is no intuitive scheme for a-priori identifying motions with strongly similar submotions, we did not explore this question further. To answer your question directly, if one wanted to, one could reuse the learned dimensionality reduction for a new task which you believed to be similar, or copy certain components of a previous motion's GMM into the new GMM and leave them fixed throughout the optimization.

10. *"The experiments have the same task-space goal as the demonstrations. Can the authors include experiments that have goals different from demonstrated, especially in orientation?"*
`Response:` Yes. As we have shown in Appendix I, regardless of the target location (position/orientation), the motion generated by the proposed dynamical system asymptotically converges to the desired target. However, it is important to note that as (8) in Appendix I is semi-negative-definite and not negative-definite, there might be joint configuration (q) which result in $J(q)\sum_{k=1}^{K}\theta_k(q)A_kJ^T(q)=0$. In this situation, the robot is stuck in a deadlock and, consequently, cannot reach the desired target.

11. *"In similar light, maybe authors could present online adaptation of A(q) to generalize to situations different than demonstrated, for example a slightly moved obstacle?"*
`Response:` The reviewer has pointed to a very interesting topic and authors appreciate the reviewers suggestion. In this paper, we assumed that the learned model (namely πs, μs, Σs and As) does not need to be updated or modified during the task execution. Hence, this would be out of the scope of this paper and we are planning to address it in future works.

12. *"In table II, instead of the tracking error, maybe the more useful error to see would be the error with respect to the plan or demonstartion. One would assume that the proposed approach would be worse at it than SEDS+IK because the minimized error is with respect to the velocity and goal, not the position of the movement."*
`Response:` This is, in general, not true. Based on the robot's motion and its configuration, there might be cases where the motion generated by SEDS does not even reach the target. To clarify this, we have systematically studied the performance of the motion generated by JT-DS and SEDS+IK in terms of (i) reaching to the desired target and (ii) following the demonstrated end-effector motions, See Figure 1. We used the data-sets from the seven tasks mentioned in Section V.A to construct SEDS and JT-DS. Both methods are tested by the same the target and the initial configurations, which were extracted from the demonstrations. We define $x^*=[x^*_{ori};x^*_{pos}]$ as the target configuration that includes the desired position as well as the desired orientation of the end-effector. The motion generated by SEDS is converted into the 7-DOF joints state using the damped least square IK solver.
In Figure 1a and Figure 1b, the error between the final location of the robot and the target is illustrated. As it can be seen, the motion generated by the proposed dynamical system reaches the target, whereas, the motion generated by SEDS does not; regardless of the fact that both models are theoretically stable. The main cause of failure for SEDS is due to the IK solver. As SEDS generates the motion of the robot without considering its joint configurations, the robot might reach its joint limits. Whereas, as JT-Ds directly shapes the joints' motion, by learning the desired joint-space behavior, one can avoid these situations.
In Figure 1c and Figure 1d, the error between the demonstrated and the excused motions are illustrated. As it can be seen, even though JT-DS is not tailored for following the end-effector motions, its performance is comparably better than SEDS.
Although including this study would highlight one of the advantages of using JT-DS over Cartesian-motion-generators, as this does not highlight the main focus of this paper and due to the space limitations, we have decided to not include this in the paper.

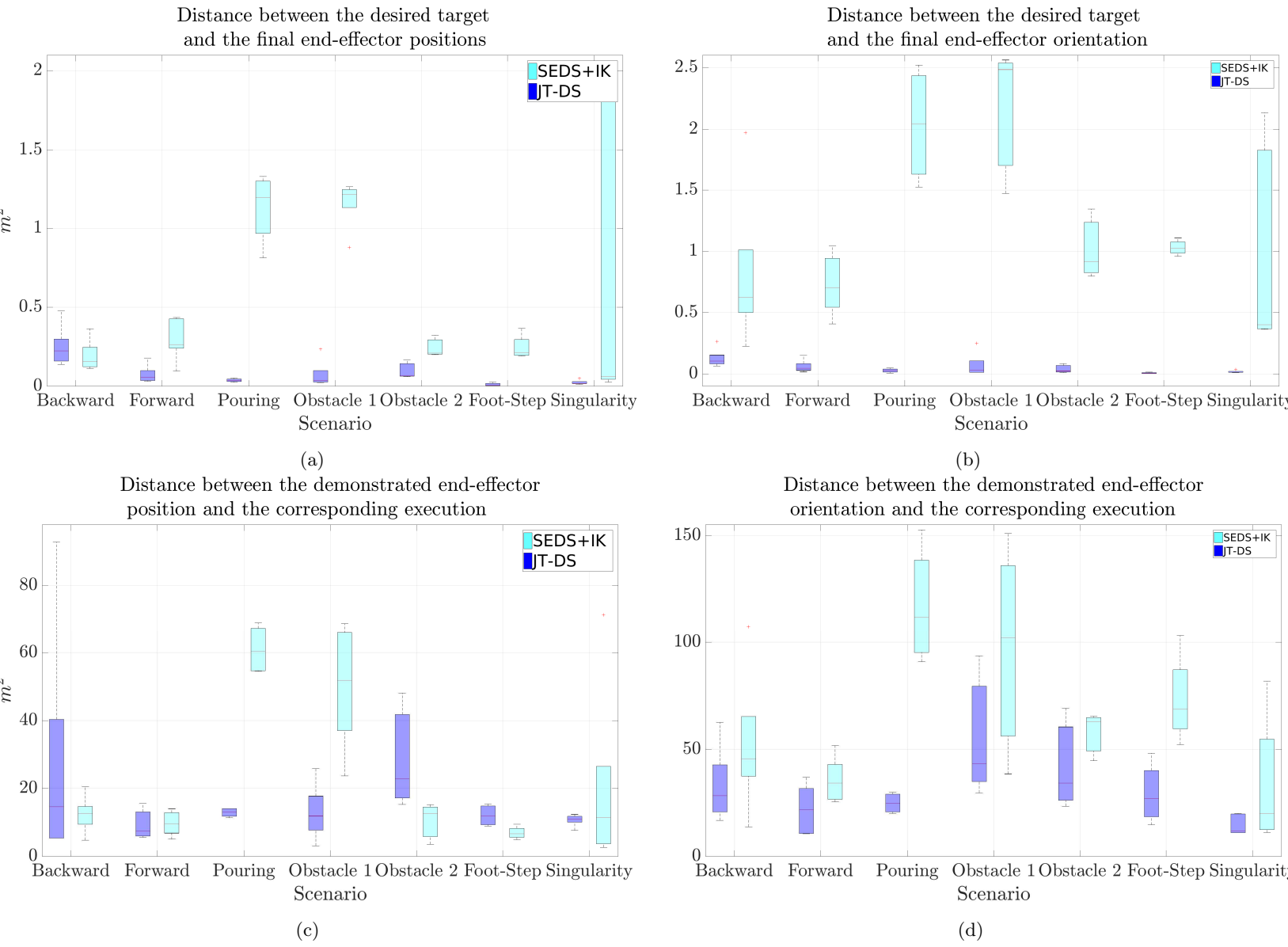


Figure 1: The performance comparison between SEDS+IK and JT-DS while the initial and the target points are same. The error between the orientations is the difference between the first two columns of the corresponding rotation matrix representation.

Comments from Reviewer #2 (Reviewer: 5 (Review30371)):

1. *"My first concern regards you choice of representing the demonstrated movement as a sequence of postures and hence looking for synergies by using PCA of K-PCA analysis. Why not using functional PCA (f-PCA) analysis and determine the few moving synergies that make possible the execution of the demonstrated behaviours? It would be that, when you combine the local synergies in space and time by using functions $\theta_k(q)$, you get what you might obtain by using directly f-PCA. However, the last one are valid for the whole movement and can be combined by constant coefficients."*
   `Response:` See below.
2. *"I also found the paper lacking of physical intuitions, especially in the experimental part, about the entities introduced in previous sections. In particular, what is the lesson we learn after your analysis of synergies and the obtained lower dimensional space for the few movements shown in section V? Is it possible to extract some global synergic behavior? Moreover, in section IV.A authors introduce the function phi(q) named the embedding that maps the joint configuration*

into a lower dimensional space. Is it associated to a whole behaviour or, as matrices A_k, has a local validity? I lost this point. Can you show/report or describe what it represents in the joint space for one or more cases in Section V? What is this lower dimensional space for the cases in Section V?"

`Response:` See below.

3. *"A similar question for the matrices A_k that represent local, I would say postural (see previous comment), synergies whose dimension is m x m, i.e. in joint space: I would expect they were defined in the lower dimensional space. Moreover, are they local representation of the distribution of postures and hence covariance matrices? What do their eigenvectors and eigenvalues locally represent? Maybe, a comparison with the classical and more known postural synergies for the human hand, analyzed by PCA, could be useful. Finally, what does it mean that they should be positive definite (apart from stability issue)?"*

`Response:` See below.

4. *"What is the dimension of function $\theta_k(q)$ that, to some extent, can be seen as the new control variables. Is it $K < m$?"*

`Response:` In response to Comments 1-4, following we mathematically and graphically analyze the proposed local behavior synergy extraction scheme which is composed of dimensionality reduction of the joint trajectories (via. PCA or K-PCA) and grouping of local behavior regions with GMM on this lower dimensional space. The K clusters found in this embedded space correspond to the number of K synergy matrices that will parametrize $\mathcal{A}(q)$. Let's begin by analyzing the proposed DS-based control-law (1). The intuition behind this control-law is that $\mathcal{A}(q) = \sum_{k=1}^{K} \theta_k(\phi(q))A_k$ is a linear combination of time-invariant linear matrices $A_k \in \mathbb{R}^{m\times m}$, these matrices are the "local behavior synergy matrices" that shape the motion in joint-space. Meaning that, given the joint-space velocity vector representing the task-space error $J^T(q)(H(q) - x^*)$, the resulting motion is biased to use a particular set of joints, defined in each A_k. Each A_k is thus activated depending on the current "joint-posture" q (represented in the lower-dimensional space $\phi(q)$) via the scheduling/activation function $\theta_k(\phi(q))$. $\phi(q)$ is a learned transformation, which is associated with a whole behavior, rather than a particular A_k (as is now clarified in IV.A). However, the precise nature of the transformation depends on the DR technique: PCA applies a uniform matrix multiplication of joint position everywhere, whereas KPCA is a more complicated function of joint position. The result of the embedding is a lower-dimensional representation of the joint position. The following line has been added to IV.A for clarity:

"For example, if the shoulder and arm joints are coupled throughout the motion, and $\phi(\cdot)$ were a matrix multiplication, it could map the "shoulder" and "arm" components of q into a single "shoulder-arm" component in $\phi(q)$."

Also, Section III has been considerably re-written to clarify our proposed approach and proposed definition of "local behavior synergies".

Continuing with the analysis, given the raw demonstrations of a joint-space behavior with a task-space target as shown in Fig. 2 (for the pouring task) we seek to learn a linear combination of local behavior synergies $\mathcal{A}(q) = \sum_{k=1}^{K} \theta_k(\phi(q))A_k$ such that we can use (1) to accurately reproduce these demonstrations. In order to do so, we project the raw joint trajectories to a lower-dimensional space via $\phi(q)$, in the case of PCA $\phi(q) = M_p \times q$ for $M_p \in \mathbb{R}^{p\times m}$. By choosing the first 3 PCs we can explain 95% of the variance in the joint posture trajectories, which can be represented in this 3-D space as shown in Fig. 2. We then jointly learn the activation function $\theta_k(\cdot)$ and the number of K local synergy regions by fitting a GMM to this representation of the joint posture trajectories. As can be seen in Fig. 2, the entire joint-space motion can be represented by 3 local synergy regions in PCA-space. Meaning that we only need 3 synergy matrices A_k to represent the complex joint-space motion accurately. The activation function $\theta_k(\cdot)$, as described in the mansucript, is the posterior probability of a joint posture in PCA-space $\phi(q)$ belonging to one of the local synergy regions $k \in \{1, \ldots, K\}$. As mentioned in Sec. V.A., we treat K as a hyperparameter and thus choose it differently for each learned motion. For motions that do not vary much throughout the course of the motion, smaller K might be sufficient, but K can just as easily be greater than m (though as K increases, so does the learning problem's computational cost). If comment-4 was about each specific $\theta_k(q)$, they are scalars, as is mentioned in Sec. III.

As mentioned in the reviewers 3rd comment, since we have an accurate representation of our joint posture trajectories and the local synergy regions in this lower-dimensional space, why not learn the A_k matrices in this space; i.e. $\mathcal{A}(q) \in \mathbb{R}^{p\times p}$ rather than $\mathcal{A}(q) \in \mathbb{R}^{m\times m}$? This seems like the obvious method, if we were considering our synergistic approach the classical way, i.e. using the lower-dimensional space as the new control variables. However, if we were to do that, this would mean that (1) would become:

$$\underbrace{\dot{q}}_{\mathbb{R}^{m\times 1}} = -\underbrace{M_p^{-1}}_{\mathbb{R}^{m\times p}}\underbrace{\mathcal{A}(q)}_{\mathbb{R}^{p\times p}}\underbrace{M_p}_{\mathbb{R}^{p\times m}}\underbrace{J^T(q)}_{\mathbb{R}^{m\times d}}\underbrace{(H(q) - x^*)}_{\mathbb{R}^{d\times 1}} \tag{2}$$

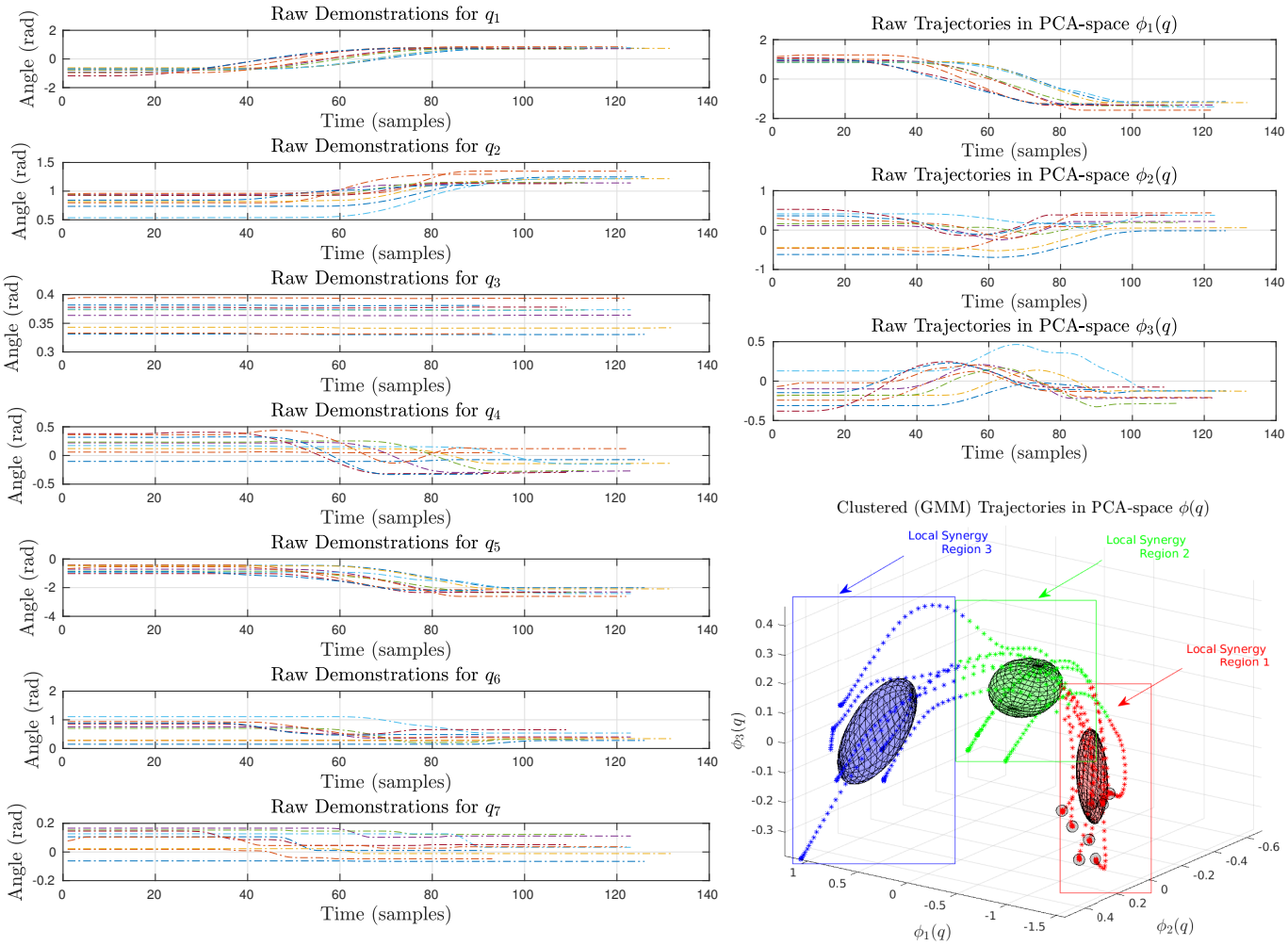


Figure 2: Raw demonstrations of the pouring task in original space, PCA space and the extracted local synergy regions.

This is not entirely incorrect, however, with such a control law we can no longer guarantee stability and convergence to the attractor x^*. By considering the Lyapunov function used in Appendix I to prove stability of (2) i.e. $V(q) = \frac{1}{2}(H(q) - x^*)^T(H(q) - x^*)$, the derivative of V wrt. would be:

$$\begin{aligned}\frac{dV(q)}{dt} &= (H(q) - x^*)^T J(q)\dot{q} \\ &= -(H(q) - x^*)^T J(q) M_p^{-1} \mathcal{A}(q) M_p J^T(q)(H(q) - x^*) \\ &= -(H(q) - x^*)^T J(q) \underbrace{M_p^{-1}\Big(\sum_{k=1}^{K} \underbrace{\theta_k(q)}_{\succ 0} \underbrace{A_k}_{\succ 0}\Big) M_p}_{\text{This should be } \succ 0} J^T(q)(H(q) - x^*) \leq 0\end{aligned} \tag{3}$$

In order for (2) to be asymptotically stable wrt. the attractor x^* the term $M_p^{-1}\big(\sum_{k=1}^{K} \theta_k(q) A_k\big) M_p \in \mathbb{R}^{m \times m}$ should undoubtedly be $\succ 0$. Even if $\theta_k(q) > 0$ and $A_k \succ 0$ for all k, the only way for this condition to hold is if M_p where a full symmetric positive definite matrix. However, this is not the case because $M_p \in \mathbb{R}^{p \times m}$ is a projection matrix, i.e. $p < m$. The term $M_p^{-1}\big(\sum_{k=1}^{K} \theta_k(q) A_k\big) M_p$ will no longer be full-rank, it will preserve the p positive eigenvalues from the A_k matrices, but the remaining eigenvalues will be 0, thus generating a semi-positive definite matrix, which invalidates our stability proof. Note that, as described in the Appendix, $J(q)$ is not proven to be full rank in all regions of the workspace, when this happens, this means that the end-effector pose is not manipulable. Hence, we can only

prove asymptotic stability when the end-effector pose is manipulable. This is a physical artifact from the joint-space constraints of the robot arm which we cannot avoid. Yet, since the demonstrations are acquired via kinesthetic teaching we can assume that they are manipulable.

As asked by the reviewer, what does it physically mean to have all positive eigenvalues? When $M_p^{-1}\big(\sum_{k=1}^{K}\theta_k(q)A_k\big)M_p$ or $(\mathcal{A}(q))$ has some eigenvalues $= 0$ the physical meaning behind it is that we are forfeiting the control of these directions of motion in joint-space. Namely, the desired velocity directions in joint-space will be constrained in the corresponding DOF of the arm, this will prohibit us from converging to the attractor and will be no longer capable of accurately following the joint-space demonstrations. If instead we learn the synergy matrices in joint-space following (1) and modulate them with the activation function in PCA-space, we avoid all of this problems and can ensure full controllability of the motion in joint-space.

Remark 1: This analysis is only considering PCA as the DR approach. If we would consider K-PCA, deriving such a control law is more involved as the inverse mapping $\phi(q)^{-1}$ is not straightforward if at all possible, thus limiting the approach to linear DR techniques. By considering $\phi(q)$ solely in the activation function $\theta(\cdot)$ we do not have this problem and can use any DR technique, as long as we're able to compute out-of-sample projections.

Remark 2: If we were to bypass the DR step our state-dependent system matrix would become $\mathcal{A}(q) = \sum_{k=1}^{K}\theta(q)A_k$, which (via our definition) is also a combination of "local behavior synergy matrices" with the activation function living in joint-space rather than the PCA space; better-known as the *joint postural synergy space*. The fact that we use PCA/K-PCA to represent the activation function $\theta(\cdot)$ is to alleviate the complexity of extracting the local synergy regions in joint-space directly. It is, indeed, also motivated by the fact that we know that multi-DOF joint postures can be accurately represented in a lower-dimensional space composed of the principal components of the postural trajectories; i.e. the well-known *joint postural synergies*. As shown in TABLE I of the manuscript and discussed in Section V, by using a lower-dimensional manifold to represent the joint trajectories, we are getting rid of outliers, noise and redundancies that might arise from the raw joint demonstrations. This is explicitly highlighted in Section V:
"As can be seen, for all datasets there is a significant increase in performance on the testing sets when using either dimensionality reduction (DR) approaches. This suggests that using DR to encode our activation functions θ_k in a lower-dimensional space $\phi(q)$ yields better generalization capabilities than encoding the behaviors in using the original q. This is most notable for the three pouring motions, where the joint-velocity RMSE testing error for a JT-DS model learned without DR is an order of magnitude higher than with DR. Such an error indicates that the demonstrated joint-behavior was over-fitted on the training set, which is also exhibited in the higher number of K needed to represent the motion without DR. For all datasets, the DR methods provided $\delta < d/2$, either comparable or less number of local behaviors synergies K and better RMSE errors on testing sets as opposed to no DR. By finding a lower-dimensional manifold to represent the joint trajectories, we are getting rid of outliers, noise and redundancies that might arise from the raw joint demonstrations. Hence, through DR we are capable of robustly extracting the local behavior synergies from raw demonstrations."

Hence, through DR we are capable of robustly extracting the local behavior synergies from raw demonstrations and learn a *time-invariant* controller that ensures reaching the task-space target while approximately following the demonstrated motion, as shown in Fig. 3, where we plot the joint positions/velocities of 1 demonstrations of the pouring task and the joint-positions/velocities from the JT-DS controller. In these plots, we are taking the velocity generated directly by JT-DS. The joint-positions are then computed by numerical integration of these velocities. Note that our controller does not generate this entire trajectory in one-shot, it is generating the next desired velocity given the current joint posture, e.g. $\dot{q}_{t+1} = \mathcal{A}(q_t)J^T(q_t)(H(q_t) - x^*)$ with q_t being the current joint configuration. Moreover, our controller is state-dependent only; i.e. it has no dependence on time or the duration of the desired trajectories. Hence, it can generate motions similar to the demonstrated ones at different frequencies and time durations.

We have added clarifications in the text such that no other reader gets confused with the new concept of behavior synergies that we propose as opposed to the classical joint posture synergies.

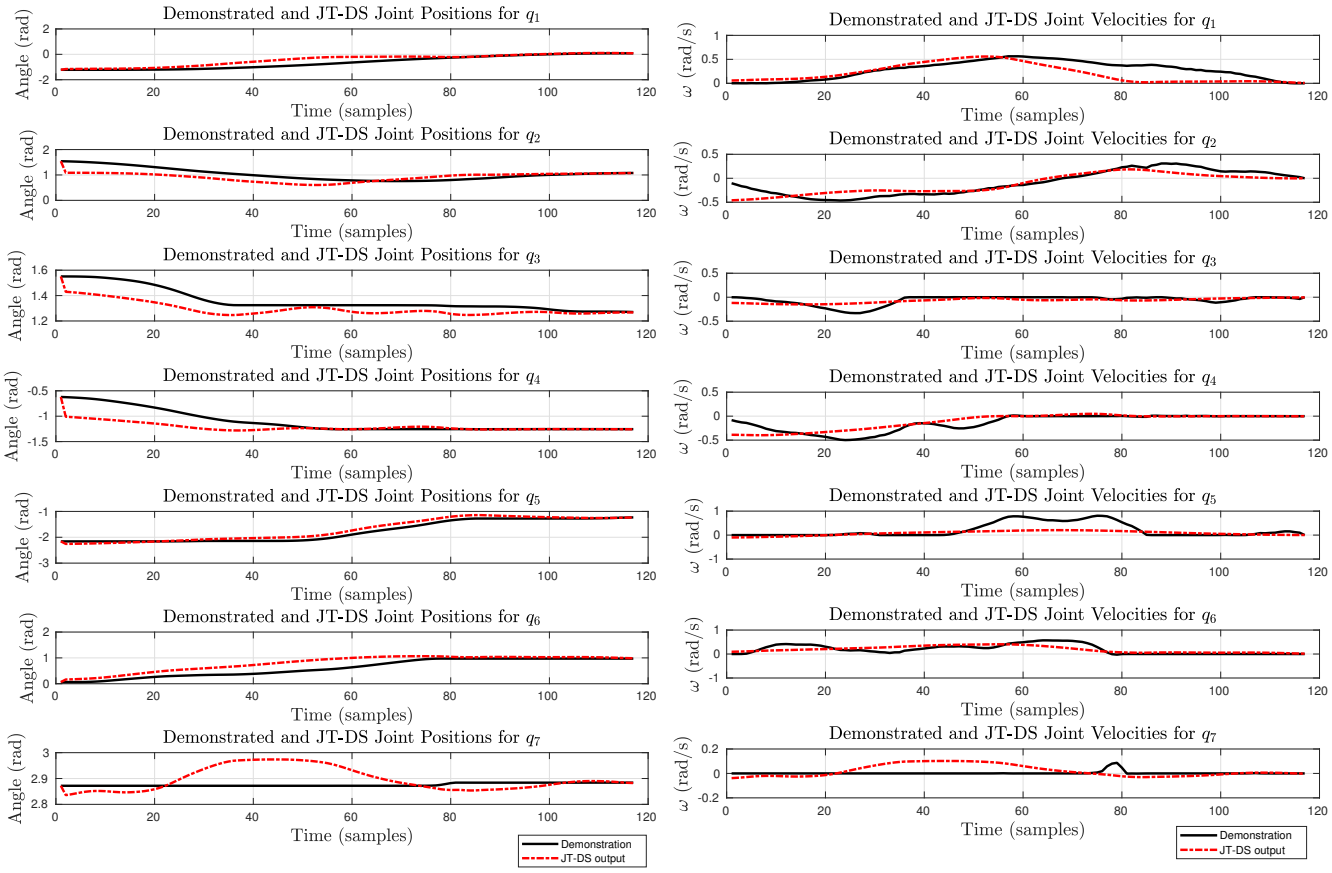


Figure 3: Training and executed joint-position/velocities with JT-DS for 1 pouring demonstrations.

On the applicability of F-PCA to our learning scheme: If we were to ignore our proposed control scheme and try to reconstruct the trajectories from the joint postural synergy components; i.e. using PCA, we would get the trajectories shown in Fig. 4. In these plots, we are taking one of the collected demonstrations projected in the PCA-space, i.e. $\phi(\{q_1, q_2, \ldots, q_T\})$ and reconstructing it to generate the desired joint posture trajectories. To compute the velocities we apply numerical differentiation on the reconstructed joint positions. As can be seen, we are capable of generating similar trajectories, both in position and velocity. In this simple example we are saving the entire demonstration in PCA-space, however, if we were to use this for controlling the real robot we need to learn a model or a controller in this low-dimensional space and then map it back to joint-space. As discussed earlier, this is not feasible with our proposed control-law (1), due to the stability and convergence objectives we defined.

Ignoring this limitation, one of the main draw-backs of this type of synergistic control is that the extracted representation in PCA space does not take into account the dynamic features of the motions and further assumes that the demonstrations have no time correlation. As mentioned by the reviewer and applied in [2] for modeling grasping motions and in [1] for upper-limb motions, functional-PCA is a nice alternative to include these dynamic features and further reconstruct entire trajectories with a linear combination of a few *time-dependent* functions. In this approach each demonstrated trajectory (of a single DOF) is considered as a function composed of a linear combination of N basis elements (which can ve exponentials, splines, Fourier) $f_i = \sum_{n=1}^{N} \theta_n b_N$. Thus, each i-th function (i.e. i-th demonstration) can be described by the vector a coefficients $\Theta = \{\theta_1, \ldots, \theta_N\}$. PCA is then performed on $\Theta = \{\theta_1, \ldots, \theta_N\}$ and the extracted PC are transformed to *functional PCs* via the N basis functions. Then $M < N$ *functional PCs* are used to reconstruct the i-th function (i.e. i-th demonstration) as $\hat{f}_i = \bar{f}_i + \sum_{i=1}^{M} c_i \xi_i$, where $\bar{f}_i$ is the mean function, c_i are the coefficients and ξ_i are the *functional PCs*. As shown in [1] with only 1 *functional PCs* 60/70% of the demonstrations can be accurately reconstructed. This approach would be interesting if our goal were to reconstruct entire time-dependent trajectories as they do in [2] and [1]. This is, however, NOT our goal and applying this method

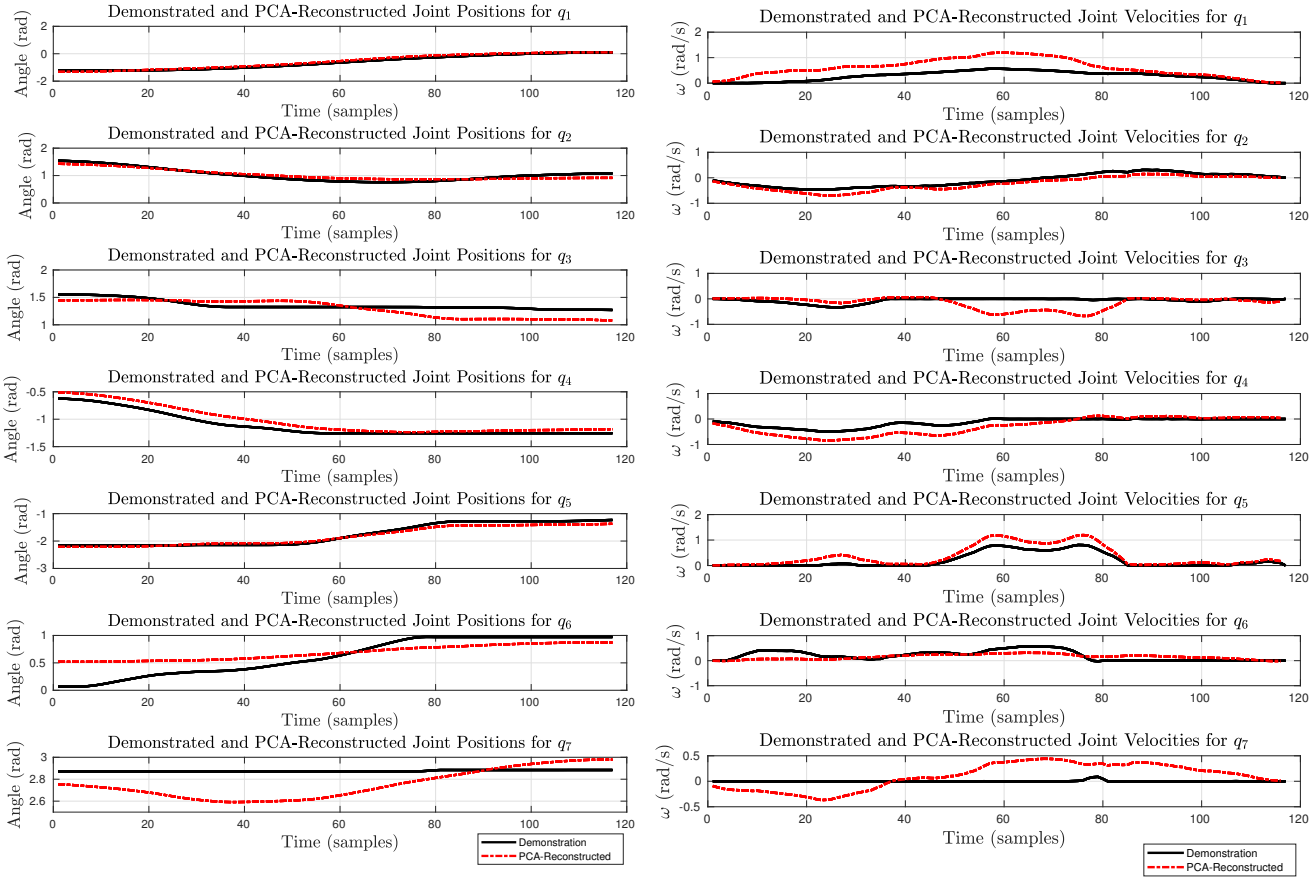


Figure 4: Training and reconstructed joint-position/velocities with PCA for 1 pouring demonstrations.

to our control scheme seems not suitable and is rather an alternative approach for the following reasons: First of all, in this work we propose a DS-based controller, meaning that, we do not generate the full reference trajectory of the robot *a prior* and then try to track it, we instead compute the next desired velocity on-the-fly which happens to result in the desired motion because it is encoded in $\mathcal{A}(q)$. Second, if we were to apply this method as the reviewer suggests, the only way to do it is to re-write a controller that uses 7 reconstructed functions $\hat{f}_i$ (i.e. one per joint) to track a time-indexed reference joint trajectory. This is a valid approach, however, in this work we propose a state-dependent; i.e. time-invariant approach. Finally, our goal not to accurately reproduce the demonstrations but rather to reach the target x^* with a similar joint motion as the one that was demonstrated.

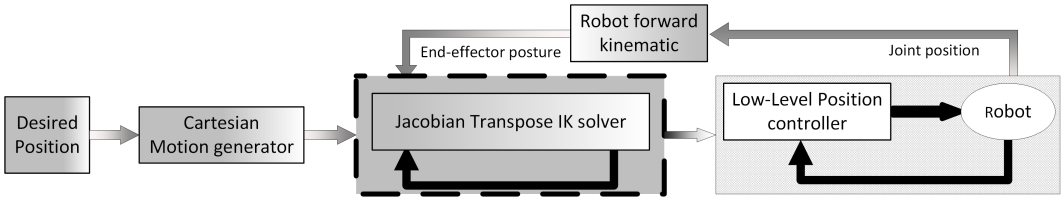
5. *"For the test, why not comparing also with the JT control method? In other terms, is it possible to use a SEDS + JT, especially for the third case "Transiting through singular configurations"?"*
Response: This could be done, but it is mathematically incorrect. The original JT control method is a DS that converges to a unique equilibrium point, i.e. the task-space target $x^d \in \mathbb{R}^{N\times 1}$, with a gain matrix $K \in \mathbb{R}^{M\times M}$ as:

$$\dot{q} = KJ(q)^T(H(q) - x^d) \tag{4}$$

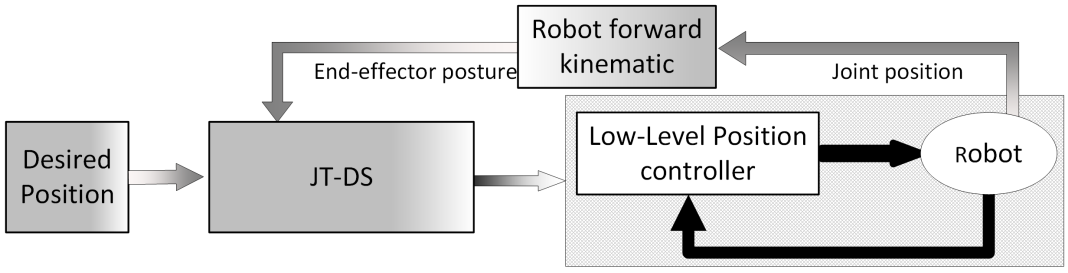
SEDS, is a DS-based motion generator that converges also to a unique equilibrium point, as:

$$\dot{x} = \mathbf{f}(x) \rightarrow \quad \Big\{ \lim_{t\to\infty} \|x - x^*\| = 0 \tag{5}$$

where $\mathbf{f}(\cdot) : \mathbb{R}^M \rightarrow \mathbb{R}^M$ is a continuous differentiable vector-valued function representing a non-linear DS that converges to a single stable equilibrium point x^*. In order to prove the stability of SEDS, one needs to assume that the dynamic of the robot is perfectly compensated by the built-in position-level controller. In other words, the transfer function of the dynamic of the robot+ the low-level controller and the IK solver must be one.
Almost all the commercially available robots are equipped by a very precises joint position controller. Hence, the



(a) A Cartesian motion generator (e.g., SEDS) with the Jacobian transpose IK solver.



(b) JT-DS.

Figure 5: A simplified schematic comparing the standard control architecture using Cartesian dynamical systems with the Jacobian transpose IK solver (top) and our propose approach. By integrating the "inverse kinematic (IK) solver" and "motion generator" blocks into a unified block, the proposed approach eliminates undesirable effects of the IK solver.

transfer function of the robot+ the low-level controller can be assumed one. Whereas, the transfer function of the Jacobian transpose IK solver can not be assumed one as the JT control method assumes that the target is static. Hence, in order for the JT controller to actually converge, the output of (5) must seem constant for a period of time. This is contradicting as (5) is always trying to converge to x^*. Thus, merging these two approaches is theoretically ill-fitting. Note that, To better highlight this, we show an illustration of the supposed SEDS+JT and JT-DS control loops in Figure 5a and Figure 5b, respectively.

6. *"Minors: - SEDS + IK: what does SEDS mean?"*
   `Response:` As stated in the previous comment, SEDS, short for Stable Estimator of Dynamical Systems is an approach proposed in [3] that learns a stable estimate of a DS with the form of (5) from demonstrations. The first paragraph of Section V.B was written to describe its meaning and how we use it:
   "We now seek to elucidate the distinctive properties of our JT-DS model by comparing its performance to a DS-based Cartesian motion generator + IK solver approach for behaviors (5-7). For the Cartesian motion generator we use the SEDS (Stable Estimator of Dynamical Systems) approach [8] which learns an asymptotically stable DS in Cartesian space from demonstrations. We then generate joint trajectories through a damped least square IK solver. From herein we refer to this approach as SEDS+IK."

Comments from Reviewer #3 (Reviewer: 7 (Review30521)):

1. *"a considerable part of the introduction is spent in comparing joint space trajectory generation to task space trajectory generation. Since these are elementary concepts in robotics, the authors might consider reducing this part"* `Response:` Several sentences in the introduction have been removed/reduced.

2. *"one of the main claims of the paper is the stability proof, which in fact is cited in the first line of the abstract. However the stability analysis is a quite simple extension of the stability of the transpose Jacobian scheme, where the constant gain on the task space error is replaced by the matrix A(q). As such the emphasis on this result might be to some extent reduced. By the way, the proof works only if the Jacobian is full rank, which is not properly discussed in Appendix I"*
   `Response:` Removed a couple mentions of the stability result, and added the dependence on the rank of the Jacobian to both the appendix and the main body of the text.

3. *"the meaning of variables q_d in (2) is not very clear to me, a well as whether it takes the same meaning as $q_{t;n}$ in (8)"*
   `Response:` q_d are the velocities from the demonstrations, which we are aiming to recreate. Equation (8) has been updated to remove the typo (they are in fact the same).

4. *"Section III uses the concept of synergy without actually defining it. An introduction to this concept is given later, in Section IV"*
   `Response:` Added the following to Section 3:
   "Each "synergy" is a joint-space transformation that biases the resulting motion to use particular joints; for more, see Sec. IV."

5. *"the expression "flow of motion" in Proposition 1 is not very clear to me"*
   `Response:` Changed to "motion".

6. *'some details about the optimization problem in (8) might be given (nature of the problem, complexity, feasibility)."*
   `Response:` The following has been added near Eq. (8):
   "The resulting optimization minimizes the mean squared joint velocity error from (**??**)*, resulting in a convex semidefinite optimization with a quadratic objective..." "This optimization is always feasible, the only constraint is that the A_ks be independently PSD."*

References

[1] Giuseppe Averta, Cosimo Della Santina, Edoardo Battaglia, Federica Felici, Matteo Bianchi, and Antonio Bicchi. Unvealing the principal modes of human upper limb movements through functional analysis. *Frontiers in Robotics and AI*, 4:37, 2017.

[2] W. Dai, Y. Sun, and X. Qian. Functional analysis of grasping motion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3507–3513, Nov 2013.

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