Surgical Procedure Understanding, Evaluation, and Interpretation: A Dictionary Factorization Approach

Abed Soleymani, Xingyu Li, *Member, IEEE*, and Mahdi Tavakoli, *Senior Member, IEEE*

***Abstract*—In this study, we present a novel machine learning-based technique to help surgical mentors assess surgical motion trajectories and corresponding surgical skills levels in surgical training programs. The proposed method is a variation of sparse coding and dictionary learning that is straightforward to optimize and produces approximate trajectory decomposition for structured tasks. Our approach is superior to existing stochastic or deep learning-based methods in terms of transparency of the model and interpretability of the results. We introduce a dual-sparse coding algorithm which encourages the elimination of redundant and unnecessary atoms and targets to reach the most informative dictionary, representing the most important temporal variations within a given surgical trajectory. Since surgical tool trajectories are time series signals, we further incorporate the idea of floating atoms along the temporal axis in trajectory analysis, which improves the model's accuracy and prevents information loss in downstream tasks. Using JIGSAWS data set, we present preliminary results showing the feasibility of the proposed method for clustering and interpreting surgical trajectories in terms of user's skills-related behaviors.**

***Index Terms*—Machine learning, dictionary learning, sparse coding, surgical trajectory decomposition, surgical skills assessment.**

I. Introduction

Modern surgical robots are capable of measuring and recording surgical activities (e.g., kinematics data, video recordings, eye-gaze data), which makes them a perfect platform for incorporating data-driven solutions for procedural understanding and user skills assessment applications. There are several work that perform the autonomous robotic surgical skills evaluation using deep learning (DL) [1]–[5]. Even though these studies have reported relatively low misclassification rates, they are black-box models in which the decision-making procedure is not transparent or understandable in human terms. As a result, there is neither intuitive nor explainable feedback to the user about his/her surgical performance and contributing factors to the predicted model outcome (in this paper, *explainability* and *intuitiveness* are referred to as the extent to which the internal mechanism or outcome of a model can be explained and are intuitive in human terms, respectively). Moreover, the high capacity of the mentioned models (i.e., the ability of the model for accommodating input data variations which mainly depends on the number of learnable parameters), especially DL models, demands large training sets to avoid overfitting. Since in the field of robotic surgery, clean and reliable data sets are very small in size, such models usually tend to overfit and fail to generate reliable models that perform well in novel situations (e.g., aborting and restarting a task, unwanted mistakes caused by poor depth estimation, etc.) [6].

To provide users with more elaborated targeted feedback about their skills level and surgical performance (e.g., in which part of the task the surgeon should improve his/her skills), one can break up surgical trajectories into pre-defined segments called *surgemes* [7] and apply surgical skills assessment methods at the sub-task level. In this paradigm, rather than having a global performance score, we will analyze the surgical workflow that results in high-resolution feedback regarding different aspects and parts of the whole executive task. There is a rich body of literature trying to perform fine-grained analysis of surgical activities in an automatic way using DL [8]–[13], reinforcement learning (RL) [14], and Hidden Markov Models (HMMs) [15]. These approaches not only have the same problems raised from being a black-box model but also suffer from over-segmentation (i.e., predicting numerous insignificant action boundaries) and low accuracy rate that prevents them to make certain predictions, especially for rare or unseen events (e.g., mid-task failures) [7]. We infer that the over-segmentation problem arises from the fact that these models pay too much attention to the data local variations (i.e., microscale details), rather than global context. Moreover, to evaluate the segmentation accuracy of these methods, they heavily rely on manually made gesture annotations, which would be very laborious, time consuming, and prone to inter-annotator variation. More importantly, the mentioned approaches break up trajectories into segments without offering any explicit interpretation about the behaviors and the dexterity of the user in the sub-task level.

Statistical machine learning (ML) techniques, on the other hand, usually perform better than DL models on small training

Manuscript received February 12, 2022; revised March 30, 2022; accepted April 14, 2022. This article was recommended for publication by Associate Editor P. Poignet and Editor P. Dario upon evaluation of the reviewers' comments. This work was supported in part by the Canada Foundation for Innovation (CFI); in part by UAlberta Huawei-ECE Research Initiative (HERI); in part by the Government of Alberta; in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada; in part by the Canadian Institutes of Health Research (CIHR); and in part by the Alberta Economic Development, Trade and Tourism Ministry's grant to Center for Autonomous Systems in Strengthening Future Communities. *(Corresponding author: Abed Soleymani.)*

The authors are with the Electrical and Computer Engineering Department, University of Alberta, Edmonton, AB T6G 2R3, Canada (e-mail: zsoleymani@ualberta.ca; xingyu@ualberta.ca; mahdi.tavakoli@ualberta.ca).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMRB.2022.3170210, provided by the authors.

Digital Object Identifier 10.1109/TMRB.2022.3170210

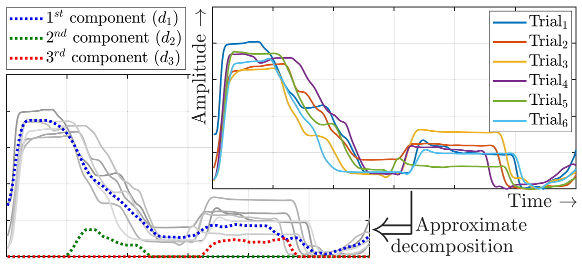

Fig. 1. Approximate trajectory decomposition in structured tasks.

data sets. An intuitive and human-inspired ML approach can provide us with a more transparent and explainable solution for data-scarcity surgical decomposition task [16]. One intuition about *structured* tasks such as suturing trials, human walking cycles, parallel parking, etc., is that their main variations can be decomposed into finite components namely *general trends* and *seasonal patterns*. This concept is illustrated in Fig. 1 in which $\boldsymbol{d}_1$ is the average general trend and other components are the averaged dominant seasonal patterns for all 6 trials within the training data set. In this kind of decomposition, each component contains specific and important intra-trial temporal variations. Moreover, taking Fig. 1 as an example, each trial in training or test data sets can be individually reconstructed via a linear combination of components, i.e., $\text{Trial}_k = \sum_{i=1}^{3} c_{ki}\boldsymbol{d}_i$ where $c_{ki} \in \mathbb{R}$ is the weight indicating the contribution strength of component $\boldsymbol{d}_i$ in the reconstruction of k^{th} trial. If $\boldsymbol{d}_i$ are trained on expert trajectories, gains c_{ki} for a given test trial can be used to determine the fidelity of the participant to the averaged *ideal* trend and seasonal patterns and convey important information regarding the execution quality and dexterity of the user. That is, each trajectory can be characterized by its coefficient vector $\boldsymbol{c}_k = [c_{k1}, c_{k2}, c_{k3}]^\top$ in the *embedding space* in which hidden behaviors of the user will be revealed.

Inspired by these intuitions, *dictionary learning* and *sparse coding* [17] can be modified to be applied on structured time series for meaningful, interpretable, and human-inspired trajectory decomposition task. Generated components (i.e., dictionary *atoms*) and their contribution in reconstruction of each individual trajectory (i.e., generated *code* matrix) disclose important information for several downstream tasks such as skills assessment, skills transfer, and anomaly detection.

In this research, we will introduce *dual-sparse* dictionary learning approach for the approximate trajectory decomposition of structured tasks in retrospective studies. We will also introduce our dual-sparse dictionary learning algorithm with a novel absolute mutual incoherence metric μ^+. The idea of *floating atoms* will be incorporated in the proposed algorithm to accommodate trajectory structures with temporal shift. This preserves the relative temporal structure of the underlying events and prevents information loss while mapping the data to lower dimensional space (or embedding space, e.g., three dimensional space which is perceptible for humans and suitable for data visualization purposes) and makes our embedding representation more meaningful and realistic. Finally, we will evaluate our method on basic structured robotic surgery trajectories in JIGSAWS data set for surgical skills assessment and anomaly detection tasks. We believe that our novel approach has a potentially high impact in robotic surgery, where demands for enhanced safety and explainability are extremely strong.

The paper is organized as follows: In Section II, basic concepts and motivations that lead us to our contributions will be discussed. In Section III, algorithm implementation and the idea of floating atoms will be presented. In Section IV, fundamental features of our approach and its application on JIGSAWS data set will be investigated. In Section V, several discussions about the advantages and practical details of the proposed method will be presented. Concluding remarks are provided in Section VI.

II. Problem Statement

A. Preliminaries

Sparse coding is a method of representing data vectors as sparse linear combinations of a set of basis elements called *atoms*. It is assumed that all atoms together which make *dictionary* matrix, capture main directions in the input space and have enough information for reconstructing input data. Dictionary learning algorithms try to develop a dictionary matrix that efficiently reconstructs each input data by a linear combination of the generated atoms. In this context, all generated coefficients of linear combinations are referred to as *code*.

A traditional dictionary learning framework for sparse representation optimizes the empirical loss function

$$\mathcal{L}(\boldsymbol{D}, \boldsymbol{C}) = \min_{\boldsymbol{D}, \{\boldsymbol{c}_i\}_{i=1}^n} \sum_{i=1}^{n} \left[\frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2 + \alpha\|\boldsymbol{c}_i\|_0 \right] \quad (1)$$

for the finite set of n data vectors $\boldsymbol{X} := [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$, aiming to find an optimal dictionary $\boldsymbol{D} := [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_p] \in \mathbb{R}^{d \times p}$ such that each data vector $\boldsymbol{x}_i$ can be well-approximated by a linear combination of dictionary atoms $\{\boldsymbol{d}_j\}_{j=1}^p$. The term $\|\boldsymbol{c}_i\|_0$ in (1) which is the number of non-zero elements in vector $\boldsymbol{c}_i$, encourages to minimize the number of non-zero element in code vectors $\boldsymbol{c}_i \in \mathbb{R}^p$ of the code matrix $\boldsymbol{C} := [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n] \in \mathbb{R}^{p \times n}$. The sparsity-promoting loss $\|\boldsymbol{c}_i\|_0$ encourages the coding algorithm to rely on a minimum number of the generated dictionary atoms for reconstruction which has the advantage of being simple and easy to decode, either by machine or human. The loss function (1) simply tries to make a balance between data reconstruction loss $\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2$ and sparsity-promoting loss $\|\boldsymbol{c}_i\|_0$ with a regularization parameter α.

The l_0 sparsity loss $\|\boldsymbol{c}_i\|_0$ in (1) makes the optimization an NP-hard problem with sub-optimal solution [18]. Replacing $\|\boldsymbol{c}_i\|_0$ with its l_1 convex relaxation $\|\boldsymbol{c}_i\|_1$ yields optimal and sparse solutions for codes $\boldsymbol{c}_i$ [19]. To prove why l_1 regularization term $\|\boldsymbol{c}_i\|_1$ enforces elements of vector $\boldsymbol{c}_i$ to be zero (i.e., to be sparse rather than small), we encourage the reader to see [20].

Due to the bi-linearity between the dictionary $\boldsymbol{D}$ and codes $\boldsymbol{c}_i$, (1) is still non-convex and cannot be jointly optimized with respect to dictionary $\boldsymbol{D}$ and code matrix $\boldsymbol{C}$ [21]. Since (1) is convex with respect to variables $\boldsymbol{D}$ or $\boldsymbol{C}$ when the other one is fixed, a solution to this problem is to alternate the optimization procedure between the two variables, i.e., minimizing the loss function with respect to one parameter while keeping the other one fixed. To prevent dictionary atoms to get arbitrary large in the optimization process, we should normalize each atom (i.e., $\boldsymbol{d}_i \leftarrow \boldsymbol{d}_i / \|\boldsymbol{d}_i\|_2, \ \forall i$) after each dictionary update.

Conventional dictionary learning approach usually generates more atoms than the number of data samples and creates over-complete dictionaries (e.g., $p \approx 4n$ in [22]) for the sake of reconstruction accuracy. Besides the high computational cost and memory demand of the over-complete dictionary approaches, the redundancy of the generated dictionary does not necessarily enhance the optimality and performance of the final solution. One major problem with over-complete dictionaries is the high correlation between generated atoms, which degrades the *mutual incoherence* metric defined as

$$\mu(\boldsymbol{D}) = \max_{i \neq j} \frac{\left|\boldsymbol{d}_i^\top \boldsymbol{d}_j\right|}{\|\boldsymbol{d}_i\|_2 \left\|\boldsymbol{d}_j\right\|_2}. \tag{2}$$

It is empirically observed that highly correlated atoms in over-complete dictionaries or in general, high values for $\mu(\boldsymbol{D})$ (in Section III-A we will calculate an upper bound for μ) make the sparse coding stage slow, computationally demanding, and non-optimal [23]. Tackling this problem, [24] proposes orthogonal dictionary learning method (we name it $\mathcal{L}^{\text{orth}}(\boldsymbol{D}, \boldsymbol{C})$) that is the constrained version of (1) subject to $\boldsymbol{D}^\top \boldsymbol{D} = \boldsymbol{I}_{p \times p}$ where $\boldsymbol{I}$ indicates identity matrix. Reference [24] argues that this approach yields $\mu(\boldsymbol{D}) = 0$, faster convergence, and comparable results relative to other sophisticated over-complete dictionary learning methods such as *K*-SVD [25].

B. Motivation

Prior work in surgical skills evaluation has shown that a lay observer is able to discover and rate the skillful behavior of a surgeon just by looking at his/her pre-recorded translational and/or rotational hand movement patterns with accuracy comparable to an expert surgeon [26]. A possible explanation about this interesting result is that the lay observer does not care about extreme details and microscale translations/rotations within the surgical trajectory. He/she just pays attention to the most informative temporal features within the executive task, i.e., (1) general trend, (2) seasonal patterns, and (3) unwanted random actions. Although the cognitive procedure of decision making based on these three factors is unknown, we are inspired to investigate how much the fidelity to the general trend, correct execution of each seasonal pattern, and minimum occurrence of incidental motions play a crucial role in rating the performance of the user in surgical tasks.

The central idea of this work is to generate an intuitive and universally understandable representation of surgical trajectories based on building blocks of surgery that meaningfully describes the procedural flow and highlights hidden information of a surgical task. The core intuition is aligned with the motivation of representing data as a sparse linear combination of specific atoms in sparse coding problems. In this context, each dictionary atom can be a representative of the task general trend or a seasonal pattern (i.e., surgeme) that encapsulates one important local variation of the trajectory.

Unlike prior dictionary learning algorithms that rely on generating a lot of atoms to take care of details and microscale variations of data, in this work, we develop an algorithm that selectively removes unnecessary atoms and focuses on preserving important variations within the input data to achieve an understandable and interpretable trajectory decomposition. Apart from neglecting unnecessary details and the small number of atoms, another feature that makes our representation meaningful and easy to interpret is the minimum amount of overlap between two arbitrary atoms. This is meaningful because we aim to assign each non-overlapping seasonal pattern of the structured trajectory to one dictionary atom (e.g., 2nd and 3rd components in Fig. 1). Moreover, a small overlap between dictionary atoms indicates that the action executed in a particular timestamp can be purely attributed to one dominant atom and it simplifies interpretations about the quality of the task done in that timestamp. Factorizing minimally-overlapping atoms can be thought of performing approximate decomposition for a particular structured task.

Although enforcing atoms to have zero overlap with each other during the learning process makes them more explainable, it degrades the signal reconstruction quality with poor non-smooth results and the interpretability of generated codes. Moreover, as we will show later, intensively reducing the total number of active atoms increases their overlap. This is because the algorithm tries to allocate all variations of the trajectory among currently existing atoms to minimize the reconstruction loss. There is a inter-dependency between the number of atoms, their overlap, and reconstruction loss to generate informative codes for a given set of trajectories. We call the smallest number of minimally-overlapping atoms that gives us a relatively good reconstruction the *intrinsic dimensionality of embedding space* (δ).

C. Data Set

All analysis in this work are done based on the standard JIGSAWS data set [27] collected from surgical activities of eight surgeons in three different levels of expertise (i.e., novice, intermediate, and expert) performing suturing (SU), knot-tying (KT), and needle-passing(NP) tasks on the *da Vinci* Surgical System. JIGSAWS contains three Cartesian motions along x, y, and z axes as well as 9 elements of rotational matrix $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$ for both hands of the user and also for both patient-side robotic arms. Note that, all 9 elements of rotational matrix $\boldsymbol{R}$ can be expressed as 3 angles *roll* (Φ), *pitch* (Θ), and *yaw* (Ψ) as follows

$$\Phi = \text{atan2}\left(\frac{r_{21}}{r_{11}}\right), \ \Theta = \text{atan2}\left(\frac{-r_{31}}{\sqrt{r_{32}^2 + r_{33}^2}}\right), \ \Psi = \text{atan2}\left(\frac{r_{32}}{r_{33}}\right)$$

where r_{ij} is the element in the ith row and jth column of $\boldsymbol{R}$.

III. Methodology

A. Dictionary Factorization

We will show that having the minimum overlap between two atoms is a much stricter condition compared to the orthogonality condition (i.e., minimum correlation between atoms or small value for $\mu(\boldsymbol{D})$) presented in [24]. To have a measure of overlap between two arbitrary atoms, we introduce the *absolute mutual incoherence* metric defined as

$$\mu^{+}(\boldsymbol{D}) = \max_{i \neq j} \frac{|\boldsymbol{d}_i|^{\top}|\boldsymbol{d}_j|}{\|\boldsymbol{d}_i\|_2 \|\boldsymbol{d}_j\|_2} \tag{3}$$

where $|\boldsymbol{d}_i|$ denotes the element-wise absolute value of the vector $\boldsymbol{d}_i$. Since $|\boldsymbol{d}_i|^{\top}|\boldsymbol{d}_j| \geq |\boldsymbol{d}_i^{\top}\boldsymbol{d}_j|$ holds for $\forall \boldsymbol{d}_i, \boldsymbol{d}_j \in \mathbb{R}^d$, from (2) and (3) it can be concluded that $\mu^{+}(\boldsymbol{D})$ is an upper bound for $\mu(\boldsymbol{D})$ (i.e., $\mu(\boldsymbol{D}) \leq \mu^{+}(\boldsymbol{D})$) and minimizing $\mu^{+}(\boldsymbol{D})$ shrinks $\mu(\boldsymbol{D})$. However, minimizing $\mu(\boldsymbol{D})$ does not necessarily yield reduced $\mu^{+}(\boldsymbol{D})$.

Property 1: $0 \leq \mu(\boldsymbol{D}) \leq \mu^{+}(\boldsymbol{D}) \leq 1$.

Proof: See Appendix A. ■

We argue that minimizing the total overlap between atoms promotes the reduction of $\mu^{+}(\boldsymbol{D})$. To formulate overlapping between atoms, we rewrite the dictionary matrix based on its rows, $\boldsymbol{D} := [\hat{\boldsymbol{d}}_1^{\top}, \ldots, \hat{\boldsymbol{d}}_d^{\top}]^{\top}$ where $\hat{\boldsymbol{d}}_k = [\boldsymbol{d}_1(k), \ldots, \boldsymbol{d}_p(k)]$ is the k^{th} row of the matrix $\boldsymbol{D}$ and $\boldsymbol{d}_i(k)$ is the k^{th} element of atom $\boldsymbol{d}_i$. Note, applying l_1 regularization on $\hat{\boldsymbol{d}}_k$ (i.e., minimizing $\|\hat{\boldsymbol{d}}_k\|_1 = \sum_{i=1}^{p} |\boldsymbol{d}_i(k)|$) in a quadratic objective function (e.g., $\frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2$) enforces elements of $\hat{\boldsymbol{d}}_k$ (i.e., $\boldsymbol{d}_i(k)$) to be zero which eventually reduces the total overlap between all atoms at timestamp k. This means setting $|\boldsymbol{d}_i(k)\|\boldsymbol{d}_j(k)|$ to zero for as many as possible $1 \leq i, j \leq p$, $i \neq j$, which ultimately minimizes $\mu^{+}(\boldsymbol{D})$. Note that since $\mu^{+}(\boldsymbol{D})$ normalizes each atom, reducing $|\boldsymbol{d}_i(k)\|\boldsymbol{d}_j(k)|$ is not enough; it is ideal for it to be zero. It means we have to have $\boldsymbol{d}_i(k)$ equal to zeros for a lot of all possible i for a given k (i.e., a sparse $\hat{\boldsymbol{d}}_k$ for all $1 \leq k \leq d$), which conceptually reduces the overlap between atoms at timestamp k. As a result, $\mu^{+}(\boldsymbol{D})$ is a good measure of overlap between dictionary atoms which $\mu^{+}(\boldsymbol{D}) = 1$ for a fully-overlapped dictionary and $\mu^{+}(\boldsymbol{D}) = 0$ for non-overlapping dictionary. Later, we will investigate the relationship between $\mu^{+}(\boldsymbol{D})$ and the total overlap between active dictionary atoms.

1) Objective Function Definition: Now, we define a new cost function $\mathcal{L}^{+}(\boldsymbol{D}, \boldsymbol{C})$ that tries to jointly generate sparse codes and dictionaries with minimally-overlapping atoms to reconstruct the input data $\boldsymbol{X}$

$$\mathcal{L}^{+}(\boldsymbol{D}, \boldsymbol{C}) = \min_{\boldsymbol{D}, \{\boldsymbol{c}_i\}_{i=1}^{n}} \left\{ \sum_{i=1}^{n} \left[\frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2 + \alpha\|\boldsymbol{c}_i\|_1 \right] + \sum_{k=1}^{d} \beta \left\| \hat{\boldsymbol{d}}_k \right\|_1 \right\} \tag{4}$$

where β is the regularization factor that determines the relative importance of total overlap compared to the sparsity of generated codes and lumped reconstruction error for all data vectors in $\boldsymbol{X}$.

2) Algorithm Implementation: As explained in Section II-A, for the sake of convexity we need to alternatively optimize loss function (4) with respect to one parameter $\boldsymbol{D}$ or $\boldsymbol{C}$ while keeping the other one fixed. As a result, we have two steps for minimizing our cost function: $\boldsymbol{C}$-step and $\boldsymbol{D}$-step. In $\boldsymbol{C}$-step we fix matrix $\boldsymbol{D}$ (coming from initialization or the previous $\boldsymbol{D}$-step) and do

$$\mathcal{L}_{\boldsymbol{C}}^{+}(\boldsymbol{D}, \boldsymbol{C}) = \min_{\{\boldsymbol{c}_i\}_{i=1}^{n}} \sum_{i=1}^{n} \left[\frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2 + \alpha\|\boldsymbol{c}_i\|_1 \right]. \tag{5}$$

This optimization is equivalent to the *lasso* problem [28] for which many fast algorithms exist. We assume that the function `sparseCode(`$\boldsymbol{X}, \boldsymbol{D}, \alpha$`)` gets the dictionary $\boldsymbol{D}$, data vectors $\boldsymbol{X}$, and regularization parameter α and returns code matrix $\boldsymbol{C}$.

In $\boldsymbol{D}$-step, we fix matrix $\boldsymbol{C}$ and do

$$\mathcal{L}_{\boldsymbol{D}}^{+}(\boldsymbol{D}, \boldsymbol{C}) = \min_{\boldsymbol{D}} \sum_{i=1}^{n} \frac{1}{2}\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2 + \sum_{k=1}^{d} \beta \left\| \hat{\boldsymbol{d}}_k \right\|_1. \tag{6}$$

Due to our l_1 condition on the rows of dictionary matrix $\boldsymbol{D}$, (6) is no longer a conventional dictionary update problem with closed-form solution via coordinate descent approach. Moreover, we have two summations that make the solution formulation hard and long.

We define the reconstruction error vector for each data sample $\boldsymbol{x}_i$ as $\boldsymbol{\varepsilon}_i := \boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i \in \mathbb{R}^d$ and reconstruction error matrix as $\boldsymbol{E} := [\boldsymbol{\varepsilon}_1, \ldots, \boldsymbol{\varepsilon}_n] = \boldsymbol{X} - \boldsymbol{D}\boldsymbol{C} \in \mathbb{R}^{d \times n}$. According to the norm-2 definition, $\|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2 = \|\boldsymbol{\varepsilon}_i\|_2^2 = \sum_{j=1}^{d} e_{ji}^2$ where e_{ji} is the element in the j^{th} row and i^{th} column of matrix $\boldsymbol{E}$. As a result

$$\sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2 = \sum_{i=1}^{n} \left(\sum_{j=1}^{d} e_{ji}^2 \right) = \sum_{k=1}^{d} \sum_{r=1}^{n} \hat{e}_{rk}^2 \tag{7}$$

where $\hat{e}_{rt}$ is the element in the r^{th} row and k^{th} column of matrix $\boldsymbol{E}^{\top}$. (7) shows the trivial fact that the sum of squares of all elements in matrix $\boldsymbol{E}$ is equal to that of $\boldsymbol{E}^{\top}$. Due to the fact that $\boldsymbol{E}^{\top} = (\boldsymbol{X} - \boldsymbol{D}\boldsymbol{C})^{\top} = \boldsymbol{X}^{\top} - \boldsymbol{C}^{\top}\boldsymbol{D}^{\top}$ we have

$$\sum_{k=1}^{d} \left(\sum_{r=1}^{n} \hat{e}_{rk}^2 \right) = \sum_{k=1}^{d} \left\| \hat{\boldsymbol{x}}_k - \boldsymbol{C}^{\top}\hat{\boldsymbol{d}}_k \right\|_2^2 \tag{8}$$

where $\hat{\boldsymbol{d}}_k$ is the k^{th} column of $\boldsymbol{D}^{\top}$ (or as previously mentioned the k^{th} row of $\boldsymbol{D}$) and $\hat{\boldsymbol{x}}_k$ is the k^{th} column of $\boldsymbol{X}^{\top}$. Combining (7) and (8) results

$$\sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{D}\boldsymbol{c}_i\|_2^2 = \sum_{k=1}^{d} \left\| \hat{\boldsymbol{x}}_k - \boldsymbol{C}^{\top}\hat{\boldsymbol{d}}_k \right\|_2^2. \tag{9}$$

Substituting (9) into (6) yields a more tractable cost function for our specific dictionary update procedure

$$\mathcal{L}_{\boldsymbol{D}^{\top}}^{+}\left(\boldsymbol{C}^{\top}, \boldsymbol{D}^{\top}\right) = \min_{\boldsymbol{D}^{\top}} \sum_{k=1}^{d} \left[\frac{1}{2} \left\| \hat{\boldsymbol{x}}_k - \boldsymbol{C}^{\top}\hat{\boldsymbol{d}}_k \right\|_2^2 + \beta \left\| \hat{\boldsymbol{d}}_k \right\|_1 \right]. \tag{10}$$

Interesting point about (10) is that it tries to minimize exactly the same problem described in (5). In fact, our specialized dictionary learning problem for a given code matrix $\boldsymbol{C}$ and data set $\boldsymbol{X}$ became an sparse coding problem for the dictionary $\boldsymbol{C}^{\top}$ and data set $\boldsymbol{X}^{\top}$ with regulation parameter β. The algorithm for solving the dual-sparse coding problem is described in Algorithm 1.

The first point regarding Algorithm 1 is that unlike the traditional dictionary learning methods, in our method norms of

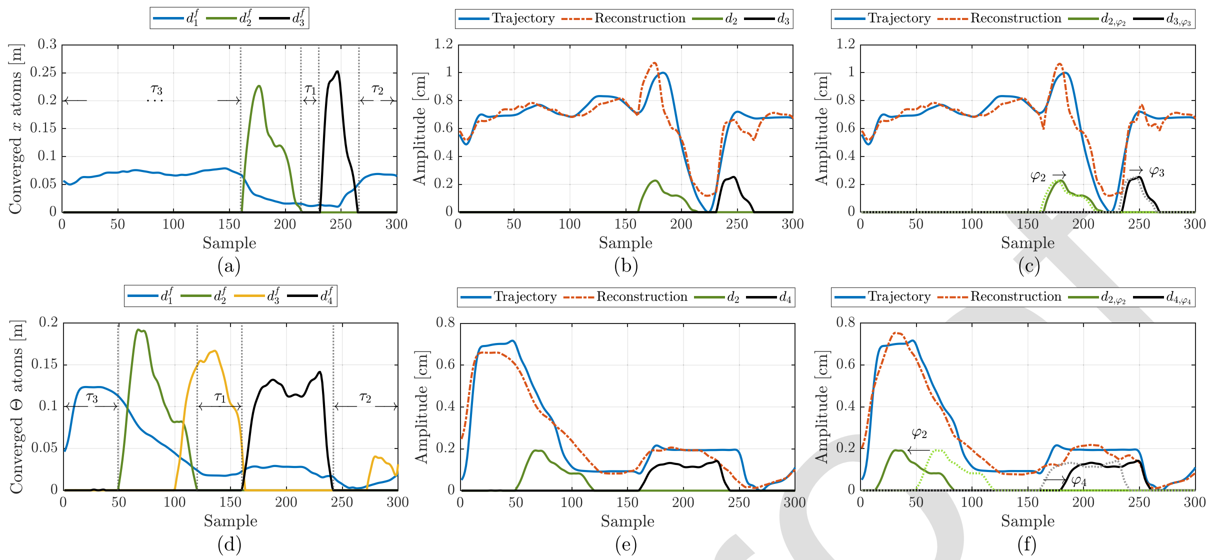

Fig. 2. The effect of temporal fine-tuning of floating atoms of a given structured task on the reconstruction and information losses, maximum possible temporal shifts (τ_i), and optimal temporal shifts (φ_i). (a) and (d) show dictionary atoms that blue lines are general trend and green and black lines are two floating atoms for that specific trajectory. (b) and (e) show bad reconstruction and information losses due to the ill temporal positioning of the floating atoms. (c) and (f) illustrate the optimal positioning of floating atoms with regards to a given trajectory.

Algorithm 1: Dual-Sparse Coding Algorithm

```
Input: $X$, $\alpha$, and $\beta$
Result: Dictionary $D$ with minimally-overlapping atoms
        and code matrix $C$
Call function sparseCode(.,.,.)
Initialize dictionary $D$
while not converged do
    $C \leftarrow$ sparseCode($X$,$D$,$\alpha$)
    $D \leftarrow \left[\text{sparseCode}(X^\top, C^\top, \beta)\right]^\top$
end
#normalizing atoms of dictionary $D$
for $i \leftarrow 1$ to $p$ do
    if $\|d_i\|_2 \neq 0$ then
        $d_i \leftarrow d_i / \|d_i\|_2$
    else
        #removing nullified atoms
        Remove $d_i$ from $D$
    end
end
$C \leftarrow$ sparseCode($X$,$D$,$\alpha$)
return $D$ and $C$
```

atoms do not arbitrary get large during the optimization process since the size of each atom is penalized by minimizing $\sum_{k=1}^{d} \|\hat{d}_k\|_1$ in (4). As a result, we do not need to normalize dictionary atoms d_i after each update. Second, we start the algorithm with initialized dictionary matrix with the number of atoms higher than the intrinsic dimensionality of embedding space of that particular task to give the algorithm the chance of deleting unnecessary atoms in its own way. In the end, the final code matrix C will be generated based on the normalized final D.

3) Algorithm Convergence: In each iteration, Algorithm 1 aims to reduce the total reconstruction loss, code sparsity loss, and dictionary atoms overlapping loss by minimizing convex cost functions (5) and (10) in C and D steps, respectively. According to [29], [30], if C^* and D^* are optimal solutions for the loss function (4) and the Algorithm 1 starts from C_0 and D_0, for each iteration $k \geq 0$ we have

$$\begin{aligned}\mathbb{E}\big[\mathcal{L}^+(D_{k+1}, C_{k+1})\big] - \mathcal{L}^+\big(D^*, C^*\big)\\ \leq \xi\big(\mathcal{L}^+(D_k, C_k) - \mathcal{L}^+\big(D^*, C^*\big)\big) \qquad (11)\end{aligned}$$

where $0 \leq \xi < 1$ is a constant derived from the properties of loss function (4) (see Appendix B) and $\mathbb{E}[\mathcal{L}^+(D_{k+1}, C_{k+1})]$ is the expected value of loss $\mathcal{L}^+(D, C)$ in $(k+1)^{\text{th}}$ iteration. (11) implies that in each iteration, statistically, it is guaranteed that the expected value of the loss function (4) approaches to its minimum value $\mathcal{L}^+(D^*, C^*)$ through the coordinate descent Algorithm 1.

B. Dictionary Temporal Fine-Tuning

Some atoms that are generated by Algorithm 1 are similar to islands with sharp onsets, short duration, and no overlap with each other (e.g., green and black atoms in the left column of Fig. 2). These atoms are the result of averaging a significant variation (i.e., surgeme) within trajectories of the training set that the algorithm perceived they are important in reconstructing the original trajectory.

An issue that may arise with these specific atoms is that they can be arbitrarily aligned with respect to the structure of a given trajectory within the test set (i.e., they can appear at different temporal positions with some phase shifts within a given executive task). In other words, although these atoms are presented in the optimal place with regards to the trajectories of the training set, their temporal position might not be optimal for a particular trajectory (either from the training set or test set). This phenomenon can neglect important variations within the original trajectory during the reconstruction and coding stage and as a result, distorts the values of the generated codes for those atoms. When the high-dimensional data (e.g., trajectory data vector) is mapped to lower-dimensional embedding space, unwanted changes in code vectors can be treated as information loss. Since the hidden behaviors of each user are mapped to the embedding space, any sort of information loss deteriorates the accuracy of any interpretation and evaluation of that trajectory.

One possible solution to this problem is to perform a post-processing step in the training stage and shift the atoms slightly to the left and right and investigate where this atom fits best (this is why we call these atoms *floating* atoms). This type of modeling due to the extra degree-of-freedom on generating atoms benefits from the advantages of over-complete dictionaries in terms of having lower reconstruction and information losses and at the same time, due to the small number of minimally-overlapping atoms avoids disadvantages of over-complete dictionary learning methods such as the high correlation between atoms discussed earlier in Section II-A. The bounded shifting to left and right is meant to secure the floating atom for intended variation within the trajectory, and to prevent it from mixing with other nearby atoms. Later we will explain one heuristic for calculating the maximum amount of shifting (i.e., $\varphi_{\max}$).

Without loss of generality, assume that first f atoms of p atoms meet all conditions of floating atoms (i.e., island-shaped, sharp onset, low duration, and no overlap with other atoms except general trend atom). A good metric to find the optimal temporal shift for a floating atom (i.e., φ_i for floating atom $\boldsymbol{d}_i$ where $1 \leq i \leq f$) is cross-correlation. Higher cross-correlation between trajectory $\boldsymbol{x}_j$ and floating atom $\boldsymbol{d}_i$ with specific time shift φ_i (we define it $\boldsymbol{d}_{i,\varphi_i}$) means that the trajectory needs the atom to be presented in that shifted position for the better reconstruction and less information loss. If the amount of cross-correlation is the same for several shifts within the domain of φ_i (i.e., $[-\varphi_{\max}, \varphi_{\max}]$), low reconstruction loss for $\boldsymbol{x}_j$ based on new dictionary $\boldsymbol{D}_{\varphi_i}$ (i.e., dictionary $\boldsymbol{D}$ with $\boldsymbol{d}_i \leftarrow \boldsymbol{d}_{i,\varphi_i}$ modification) and new generated sparse code $\boldsymbol{c}_{i,\varphi_i}$ will finalize the optimal value for shifting. The optimal shift φ_i for floating atoms $\boldsymbol{d}_i$ given trajectory $\boldsymbol{x}_j$ will be calculated by solving this optimization problem

$$\varphi_i = \arg\max_{\varphi}\left(\boldsymbol{x}_j \star \boldsymbol{d}_{i,\varphi} - \gamma \left\|\boldsymbol{x}_j - \boldsymbol{D}_{\varphi}\boldsymbol{c}_{i,\varphi}\right\|_2\right) \qquad (12)$$

where $\star$ is cross-correlation operand, γ is the regularization factor, and $\varphi \in [-\varphi_{\max}, \varphi_{\max}]$. The meta-algorithm for calculating the optimal shifting for floating atoms that can be applied to all floating atoms of a learned dictionary is presented in Algorithm 2. After applying Algorithm 2 on all floating atoms $\boldsymbol{d}_i$ with respect to the test trajectory $\boldsymbol{x}_j$ and finding φ_i for all possible $1 \leq i \leq f$, we will update dictionary $\boldsymbol{D}$ to $\boldsymbol{D}^{\dagger}$. From now on, the generated code matrix $\boldsymbol{C}_j^{\dagger}$ =`sparseCode(`$\boldsymbol{x}_j, \boldsymbol{D}^{\dagger}, \alpha$`)` will be used in downstream applications discussed in the future sections.

Each row of Fig. 2 shows one example of how floating atoms can be incorporated to reduce the information loss and enhance the accuracy of the time series mapping for a given structured task. Dictionary atoms are plotted in the left column of Fig. 2 in which $\boldsymbol{d}_1^f$ is the general trend and green and black diagrams are two atoms capturing seasonal patterns of

Algorithm 2: Meta-Algorithm for Calculating Optimal Shift φ_i for Floating Atoms $\boldsymbol{d}_i$

```
Input: $x_j$, $D$, $\varphi_{\max}$, and $\gamma$
Result: $\varphi_i$
Call function sparseCode
$\varphi_i \leftarrow -\varphi_{\max}$ #initializing optimal shift
$\mathcal{C}_{\max} \leftarrow -\infty$ #initializing optimal cost
for $\varphi \leftarrow -\varphi_{\max}$ to $\varphi_{\max}$ do
    $c_{i,\varphi}$ =sparseCode($x_j$, $D_\varphi$, $\alpha$)
    $\mathcal{C} = x_j \star d_{i,\varphi} - \gamma \|x_j - D_\varphi c_{i,\varphi}\|_2$
    if $\mathcal{C} > \mathcal{C}_{\max}$ then
        $\mathcal{C}_{\max} \leftarrow \mathcal{C}$
        $\varphi_i \leftarrow \varphi$
    end
end
return $\varphi_i$
```

the trajectory that satisfy three conditions of being floating atoms. Fig. 2(b) and Fig. 2(e) show the reconstruction quality and relative temporal positioning of each floating atom with respect to the original trajectories. As it is clear in Fig. 2(b), it is better if $\boldsymbol{d}_2^f$ and $\boldsymbol{d}_3^f$ slightly shift to the right to better capture the temporal variation that they have to represent. As shown in Fig. 2(c), optimal shift values are $\varphi_2 = \varphi_3 = 2$ samples. This change will result in 18.7% improvement of reconstruction loss and higher value for the codes assigned to these floating atoms (i.e., c_2 and c_3) that make them more accurate and meaningful in terms of reflecting skills and hidden behaviors of the user. The effect of floating atoms on Fig. 2(f) is even more noticeable. By shifting $\boldsymbol{d}_2^f$ for $\varphi_2 = 36$ to the left and $\boldsymbol{d}_4^f$ for $\varphi_4 = 19$ to the right (see Fig. 2(f)), we will achieve 27.4% improvement in reconstruction loss and considerable changes in the value of codes c_2 and c_4. For instance, the temporal position of $\boldsymbol{d}_2^f$ with respect to the given trajectory in Fig. 2(e) was too bad that the sparse coding algorithm decided to neglect this atom in reconstruction and set c_2 to zero. After fine-tuning $\boldsymbol{d}_2^f$ in 2(f), the generated code for $\boldsymbol{d}_2^f$ became $c_2 = 1.11$ which sounds more accurate and realistic in human terms.

An important objective during shifting floating atoms is to avoid merging these atoms (i.e., they should not overlap after being placed in their optimal temporal positions). For instance, in Fig. 2(a), we should have $\varphi_{\max} \leq \tau_1/2$ to avoid merging two floating atoms in their extreme shifted states. In Fig. 2(d), d_2^f and d_4^f should not move to right and left, respectively to avoid increasing their overlap with d_3^f. These objectives not only preserve the isolated nature of floating atoms but also reduce the computational cost of finding optimal values for all possible shifts. If we maintain the non-overlapping property of floating atoms while shifting them to find optimal φ_i, they will remain decoupled. As a result, we can independently find φ_i for each valid $\boldsymbol{d}_i$ instead of jointly optimizing (12) for all possible floating atoms.

Another upper bound for $\varphi_{\max}$ is the maximum amount of sliding for each floating atom until they reach left or right boundaries of the time-series (e.g., τ_2 and τ_3 for the black and green atoms respectively in the left column of Fig. 2). A possible heuristic for finding a good upper bound for $\varphi_{\max}$ based

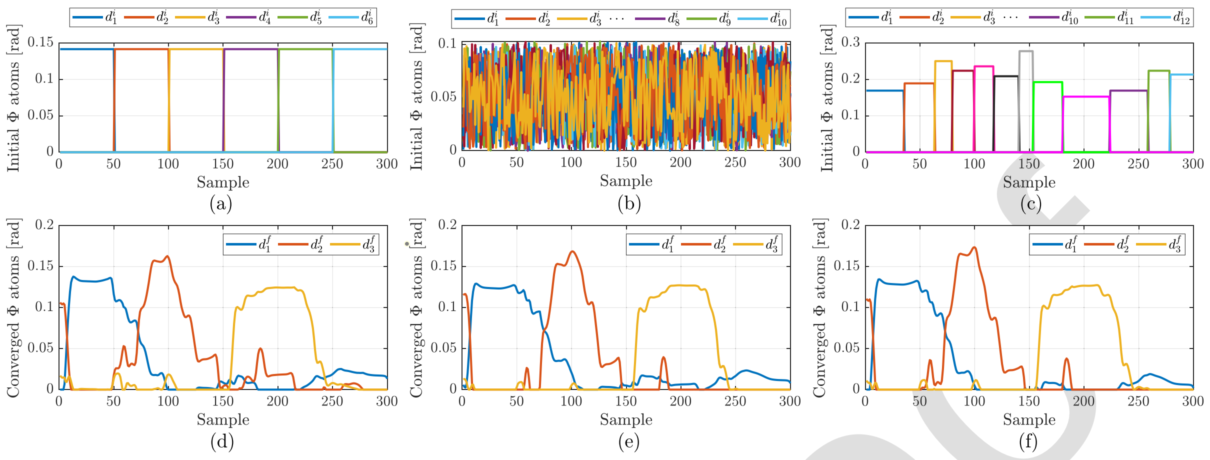

Fig. 3. The effect of different initialization of dictionary matrix $\boldsymbol{D}$ in Algorithm 1 convergence. Each column shows the initialized atoms at the top and their converged final atoms with $\delta = 3$ at the bottom. Different initializations converged to almost identical results (especially in the main variations) which indicates the consistency of the dual-sparse algorithm in finding meaningful atoms.

on explanations above and examples shown in Fig. 2(a) and Fig. 2(d) are $\varphi_{\max} \leq \min\{\tau_1/2, \tau_2, \tau_3\}$ and $\varphi_{\max} \leq \min\{\tau_2, \tau_3\}$, respectively.

Finding an appropriate value for $\varphi_{\max}$ is case-dependent and needs human investigations for getting acceptable results. Since $\varphi_{\max}$ should be calculated for each floating atom of each component within all trajectories (e.g., x, y, z, Φ, Θ, and Ψ of left and right hands), the complexity of the dictionary temporal fine-tuning is proportional to the complexity of the task (i.e., having numerous sub-tasks) and the number of components in each trajectory.

IV. Application to JIGSAWS Data Set

Inspired by the fact that decomposing surgical trajectories into their main variations (i.e., surgemes) reflects skills better than methods based on execution time or total path length [31], we resampled all surgical trials within the JIGSAWS data set to 300 samples and rescaled them between 0 and 1. According to our investigations, no data variation is removed during the resampling/rescaling since we have no sudden motion in surgical tasks. We can investigate task execution time and total path length as other two factors for our further investigations.

A. Model Training

Results presented here and in the next section are based on atoms trained over surgical data of an expert user out of 8 subjects in the JIGSAWS data set. Using expert data for training is due to the fact that global trends and seasonal patterns with minimum random movements within a particular task can be found in expert trajectories. Such a model can be used as a benchmark for other users to discover their hidden abnormal behaviors. It is worth mentioning that the processing time for different initializations and different trajectory components ranges from about 15 to 25 seconds.

B. Convergence Consistency

In a conventional dictionary learning approach the algorithm converges to very different dictionaries (i.e., non-similar local optima) for differently initialized atoms. Proper, robust, and informative initial point as an important factor for the success of the dictionary learning algorithm is an intensive field of research [32]. Due to our motivation, which is to meaningfully interpret an atom as a representative of a surgeme or sub-task of a surgical trajectory, randomly generated atoms are practically unusable even if they give us a good reconstruction.

In our setting, due to the dual-confined loss function described in (4), the small number of final atoms, and the structured nature of input time-series (that are coming from basic structured surgical tasks), Algorithm 1 tends to generate very similar atoms for different initial points for the dictionary. As it is clear in Fig. 3, different initializations with the different number of atoms p and different temporal lengths converged to almost identical atoms (especially in the main variations) with negligible difference in low amplitude parts. As we discussed in Appendix B, since (4) is *directional* component-wise Lipschitz continuous gradient, $\boldsymbol{D}^*$ and $\boldsymbol{C}^*$ in (11) are not necessarily global optima and hence, the convergence towards local optima is guaranteed in Algorithm 1. According to Fig. 3, the consistency of Algorithm 1 in converging to almost identical results for completely different initializations indicates that local optimum solutions are very close to the global optimum solution of (4).

The reasoning behind this consistency is that the cost function (4) *nullifies* unnecessary atoms (i.e., setting them to a vector of zeros) while forming residual ones to capture main variations within the training trajectories that represent surgemes and basic actions in a particular task. It is observed that Algorithm 1 tries to leave important variations of the currently nullified atom as inheritances to its neighboring active atoms to satisfy the reconstruction loss function. We argue that this behavior is the key element that makes our approach robust against the effect of dictionary initialization.

C. Reconstruction Quality

Perfect reconstruction (i.e., capturing almost all microscale details of input) is a major objective in the classic dictionary learning problem and also one of motivations for incorporating

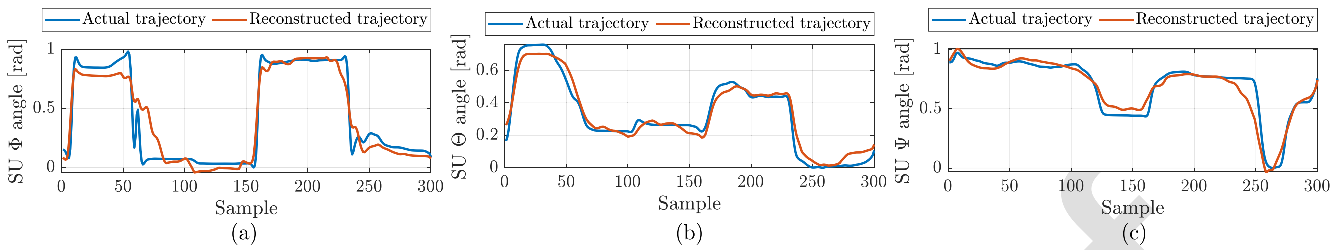

Fig. 4. Reconstruction quality of the proposed method for rotational angles of the suturing task in JIGSAWS data set. We aim to preserve main variations of trajectories while neglecting unnecessary microscale details to create explainable atoms for the decomposition task.

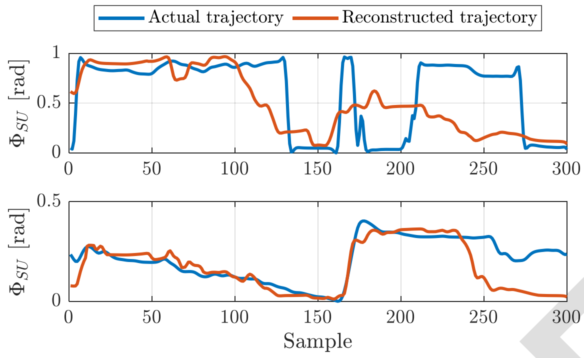

Fig. 5. The effect of fault and lack of expertise on reconstruction quality of two sample trajectories in suturing task in JIGSAWS.

over-complete dictionaries. In our setting, perfect reconstruction requires extra atoms to represent unwanted random actions within structured trajectories which increases the risk of overfitting the training set and makes it hard to interpret the cause and effect of generated codes and explain them to a human. Inspired by this fact, the motivation of our approach is to sacrifice perfect reconstruction to achieve a small number of minimally-overlapping atoms that represent prevailed ongoing surgemes within the trajectory. Following interpretations about the model reconstruction behavior in Fig. 5 indicate that this sacrifice is not in vain and helps to create an explainable approach for trajectory assessment.

Both trajectories in Fig. 5 and trajectory shown in Fig. 4(a) are reconstructed according to the atoms shown in Fig. 3(d) with hyperparameters $\alpha = 1$, $\beta = 1.5$, and $\gamma = 0.5$. The upper plot of Fig. 5 belongs to an intermediate user performing suturing trial with the code matrix of $\boldsymbol{c}_{\text{In}} = [7.33, 4.89, 3.13]^\top$ which is the same task performed by the expert user shown in Fig. 4(a) with the code matrix of $\boldsymbol{c}_{\text{Ex}} = [5.89, -0.57, 7.26]^\top$. One notable source of difference is the unusual behavior at the middle part of the intermediate trajectory that is the sign of happening mid-task failures and restarting while inserting the suture needle inside the phantom tissue. This common anomaly in surgical tasks is noticeable via bad reconstruction and will leave its trace in embedding space by distorting the code values of its neighboring atoms. In this particular case, c_2 for instance, should be a low value for normal task execution (e.g., $c_2 = -0.57$ in $\boldsymbol{c}_{\text{Ex}}$), but because of the anomaly, c_2 becomes higher than usual in $\boldsymbol{c}_{\text{In}}$. This example indicates that although the effect of random actions is not explicitly a part of problem formulation, they implicitly leave their interpretable tracks in code space.

The lower plot of Fig. 5 belongs to a novice user with the code matrix of $\boldsymbol{c}_{\text{No}} = [1.92, 0.59, 2.49]^\top$ which is significantly dissimilar to the expert trial shown in Fig. 4(a) with the code matrix of $\boldsymbol{c}_{\text{Ex}}$. For instance, one important source of the dissimilarity is the low values of roll rotation at the beginning of the trial that results in low value for code c_1 corresponding to d_1^f in Fig. 3(d). The low value of c_1 is the sign of a fundamental mistake is suturing task that will be elaborated in Section IV-D. For the sake of further clarification, a supplementary video is provided that explains the descriptions above along with the endoscopic videos of the trials plotted in Fig. 5.

D. Embedding Space Analysis

In a broader sense, the proposed dual-sparse dictionary learning approach maps the input trajectory into a lower-dimensional space (i.e., code or embedding space) that reveals the latent temporal structure of data that can be used for skills assessment, anomaly detection, and educational purposes. For the sake of clarification, embedding space of Φ angle for all suturing trials based on atoms in Fig. 3(d) trained over sample trajectories of the first expert Ex_1 is shown in Fig. 6(a) and Fig. 6(d). Similarly, embedding space of Θ angle for all suturing trials based on atoms in Fig. 2(d) trained over sample trajectories of Ex_1 is shown in Fig. 6(b) and Fig. 6(e). Finally, embedding space of Ψ angle for all suturing trials based on dictionaries trained over sample trajectories of Ex_1 is shown in Fig. 6(c) and Fig. 6(f). As it is illustrated in Fig. 6, representations of different trials of each subject cluster near to each other in the embedding space. This behavior reflects their surgical *style* in the low dimensional space. Moreover, subjects with similar skills levels usually cluster near each other. This can be useful for investigating the learning curve of a trainee while his/her latent representation approaches towards expert clusters after few sessions of training.

Another interesting point about plots in Fig. 6 is that, most non-expert subjects are pretty similar in deviating from the general trend d_1^f (i.e., $c_1 \approx 0$). This behavior results in compact near clusters along the c_1 axis for less-experienced users. However, since each non-expert user has his/her own way of deviating from the general trend this coherency does not imply consistency in performing the task. On the other hand, expert users have higher fidelity to the general trend by having large codes (e.g., $4 < c_1 < 8$ in Fig. 6(d)). However, this variation in c_1 cannot be attributed to the lack of consistency in performing the task. This is because d_1^f is normalized to one and has a relatively small maximum value compared to that

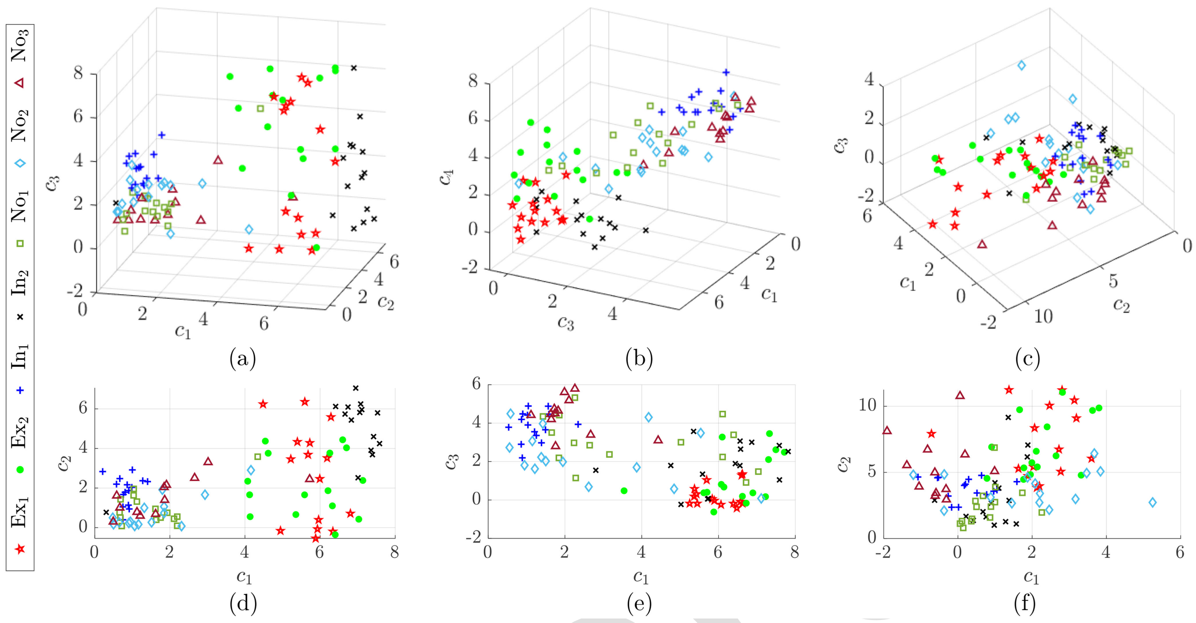

Fig. 6. Embedding space representation of three different angular rotations (from the left column to right: Φ, Θ, and Ψ) of the suturing task for two expert users Ex_i, two intermediate users In_i, and three novice users No_i trained over Ex_1 trajectories in JIGSAWS data set. For better visualization, the second row shows 2D angle of the 3D plot of the same column.

of trajectories. As a result, slight shifts or scales in trajectories may considerably change the value of c_1.

The mentioned interpretations can provide surgical mentors with good clues about the regular mistakes between beginners and help them to develop their lectures and training flowcharts accordingly. For instance, almost all novice and intermediate trainees can be confidently separated from expert ones according to their low value of c_1 in their embedding space shown in all plots of Fig. 6. In general, the low values of c_1 in the rotational data of the suturing task demonstrates the low angular dancing of the user while inserting the needle inside the tissue. This means the user mistakenly inserts the needle with translations rather than properly orienting the tool. This is mainly due to the bad needle positioning upon the surgical incision during the suture needle insertion. This is a common mistake in suturing tasks and this clue can be used as an important topic in training courses. To have a better understanding, please watch the supplementary video.

Moreover, bad needle positioning at the beginning of the task will propagate and lead to an extra Θ twist at the end. High values of c_3 in Fig. 6(e) demonstrates high pitch rotation at the end of suturing when the needle should come out of the tissue. It is the sign of inconvenient task completion due to the bad start that can harm the tissue and wrist of the surgeon.

Additionally, since $\boldsymbol{d}_1^f$ in Fig. 2(d) is corresponding to the general trend in the trajectory, low values of c_1 in Fig. 6(e) can also demonstrate that less-experienced surgeons do not follow the general trend of the operation and the trajectory is mostly based on unordered motions.

E. Trajectory Bi-Clustering

Trajectory bi-clustering is another investigation that simultaneously reveals the *type* of the executive trajectory (e.g., x, y, z, Φ, Θ, and Ψ of suturing, knot tying, and needle passing tasks) and the skills level of the user. To do so, we concatenate different input matrices and their dictionaries to create $\boldsymbol{X} = [\boldsymbol{X}_1, \dots, \boldsymbol{X}_t]$ and $\boldsymbol{D} = [\boldsymbol{D}_1, \dots, \boldsymbol{D}_t]$ for t different input types where $\boldsymbol{D}_i$ is trained over data set $\boldsymbol{X}_i$. Then we let the sparse coding algorithm to generate code matrix $\boldsymbol{C}$ for $\boldsymbol{X}$ given the *combined* dictionary $\boldsymbol{D}$. Ideally, different trajectories lie in non-overlapping subspaces and $\boldsymbol{C}$ generates codes for a given trajectory $\boldsymbol{x}_i$ based on its specifically designed trajectory and assigns zero to other non-related atoms. In other words, codes generated for a particular data type $\boldsymbol{X}_i$ is $\boldsymbol{C}_i = [\boldsymbol{0}_1^\top, \dots, \boldsymbol{0}_{i-1}^\top, \boldsymbol{D}_i^\top, \boldsymbol{0}_{i+1}^\top, \dots, \boldsymbol{0}_t^\top]^\top$ where $\boldsymbol{0}_l$ is a zero matrix with the same size as $\boldsymbol{D}_l$ for all $1 \le l \le t,\ l \ne i$. In this ideal case, the code matrix $\boldsymbol{C}$ is *block-diagonal*. Fig. 7 demonstrates the same concept for two types of inputs: x and z trajectories of knot tying task, in which the code matrix $\boldsymbol{C}$ is visualized by the colormap (i.e., warmer colors indicate higher code values.) The sparse coding algorithm tends to generate non-zero codes for a trajectory based on its specifically generated atoms which results in a block-diagonal code matrix $\boldsymbol{C}$ (dashed green rectangles in Fig. 7). When the first dashed green block ends and the second one begins, we have a decision boundary between two clusters of input types (i.e., column clustering).

Moreover, expert users' trajectories have higher *fidelity* to their specific atoms. In other words, they purely belong to the subspace of their task with minimum overlap with the subspace of other tasks and almost do not generate codes for irrelevant atoms (i.e., their coefficients are zero). As a result, investigating the rows of the code matrix $\boldsymbol{C}$ (i.e., row clustering) can give us informative clues about the skills level of the user (e.g., solid red rectangles in Fig. 7 which indicate trials done by expert users). In addition to the lack of expertise, abnormality is another cause of generating codes based on other irrelevant atoms which can be considered as random

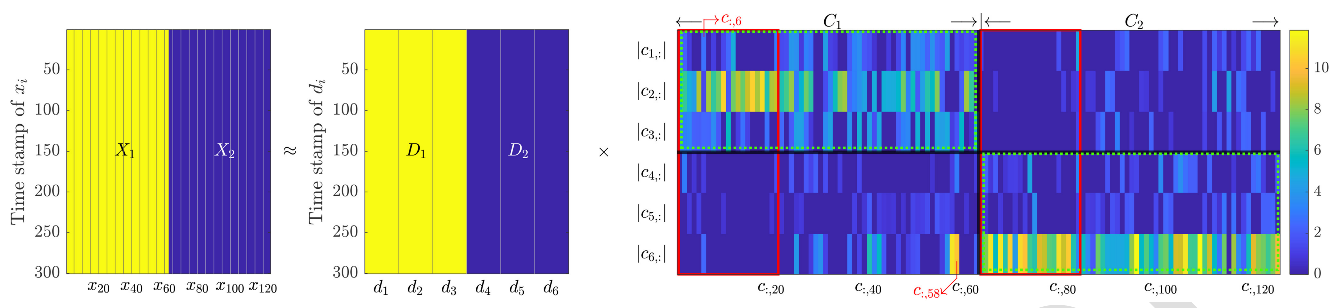

Fig. 7. Bi-clustering of x and z trajectories of knot tying task in JIGSAWS data set according to the combined dictionary $\boldsymbol{D} = [\boldsymbol{D}_1, \boldsymbol{D}_2]$ where $\boldsymbol{D}_i$ is trained over $\boldsymbol{X}_i$. The generated code matrix $\boldsymbol{C} = [\boldsymbol{C}_1, \boldsymbol{C}_2]$ for $\boldsymbol{X} = [\boldsymbol{X}_1, \boldsymbol{X}_2]$ given $\boldsymbol{D}$ is roughly block-diagonal (dashed green rectangles). We can approximately assign each code vector into four main clusters according to the task (column clustering indicated by vertical black solid line) and user skills level (row clustering indicated by red rectangles). Red solid rectangles correspond to expert trials.

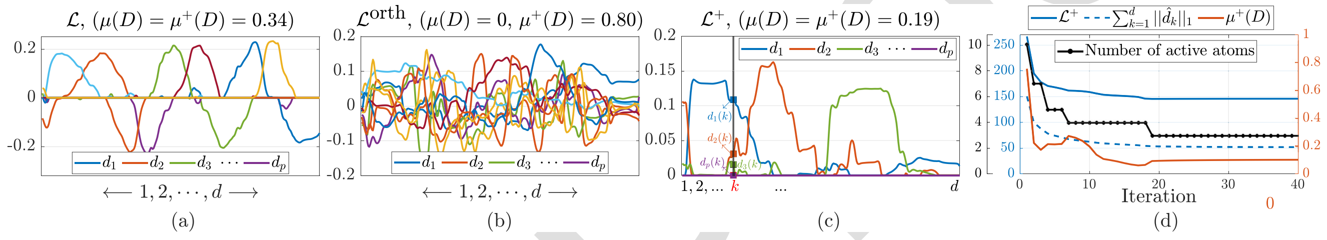

Fig. 8. Dictionary atoms generated for a given task with $p = 10$ initialized atoms based on (a) standard dictionary learning loss $\mathcal{L}$ in (1), (b) orthogonal dictionary learning loss $\mathcal{L}^{\text{orth}}$ [24], and (c) our proposed $\mathcal{L}^+$ loss described in (4) (the algorithm nullified 7 atoms and preserved the most informative 3 atoms). (d) convergence properties of $\mathcal{L}^+$ loss (each y label is associated with the plot with the same color).

atoms for that specific trajectory. As it is shown in Fig. 7, 6th and 58th trials (i.e., $c_{:,6}$ and $c_{:,58}$) have small codes for their own atoms and large codes for other non-relevant atoms. It means for both trials, something wrong is happening during the execution of that trajectory that causes these trajectories to lose fidelity to their specifically generated atoms. Further explanations and the endoscopic videos of these two trials are provided in the supplementary video.

V. Discussions

A. Advantages of $\mathcal{L}^+$ Loss

In this section, we will compare final atoms generated by our method with $\mathcal{L}^+$ loss described in (4) with atoms generated by conventional methods with $\mathcal{L}$ and $\mathcal{L}^{\text{orth}}$ losses to have a better understanding about the concepts behind these methods. As shown in Fig. 8(c), $\mathcal{L}^+$ generates optimal atoms compared to $\mathcal{L}$ and $\mathcal{L}^{\text{orth}}$ in terms of minimum overlap, reduced $\mu(\boldsymbol{D})$ and $\mu^+(\boldsymbol{D})$ metrics, and good trajectory reconstruction. A result of the conventional dictionary learning problem is provided in Fig. 8(a). In general, conventional dictionary learning method with loss $\mathcal{L}$ produces atoms with large overlaps and increased $\mu^+(\boldsymbol{D})$ compared to our approach since it does not have any penalization term for atoms' overlaps. Fig. 8(b) demonstrates atoms resulting from minimizing $\mathcal{L}^{\text{orth}}$ loss in which any pair of ten atoms have zero correlation (i.e., $\mu(\boldsymbol{D}) = 0$ or $\boldsymbol{D}^\top \boldsymbol{D} = \boldsymbol{0}$), but they have considerable overlap with each other (i.e., large value for $\mu^+(\boldsymbol{D})$). It means orthogonal atoms generated by $\mathcal{L}^{\text{orth}}$ shown in Fig. 8(b), despite their low reconstruction loss and zero correlation cannot be used for approximate trajectory decomposition for structured tasks.

Moreover, the final converged atoms based on $\mathcal{L}$ and $\mathcal{L}^{\text{orth}}$ losses highly depend on the random state for the initialization. As illustrated in Section IV, our method is robust against initialization and delivers quite consistent results.

The effect of optimizing (4) on $\mu^+(\boldsymbol{D})$ is also illustrated in Fig. 8(d). The value of $\mu^+(\boldsymbol{D})$ drops when the sparsity promoting cost for dictionary atoms (i.e., $\sum_{k=1}^{d} \|\hat{\boldsymbol{d}}_k\|_1$) decreases until the algorithm nullifies one or several atoms. When a reduction happens in the total number of active atoms, there is a mild and temporary increase in $\mu^+(\boldsymbol{D})$. This is because the algorithm is forced to assign data variations to fewer active atoms which increases the total overlap.

B. Hyperparameter Tuning

Another important feature of our method is that sparsity-promoting terms for both codes $\boldsymbol{c}_i$ and dictionary rows $\check{\boldsymbol{d}}_k$ in $\mathcal{L}^+$ loss nullifies unimportant atoms and returns the fewer number of atoms after the optimization procedure. This fact is illustrated in Fig. 8(d) by showing the total number of active atoms in each iteration which is reducing when the optimization proceeds. As explained before, this feature benefits the ease of interpretability of generated atoms. By changing hyperparameters α and β, the number of final atoms and the value of their overlap will change. Ablation study results for right hand's Φ angle (Φ_R) in suturing task in Table I, indicate that relaxing the sparsity regularization parameters α and/or β yields an increased number of final atoms and an improvement in the reconstruction loss. However, arbitrarily increasing the number of atoms by reducing α and β does not guarantee a considerable reduction in reconstruction loss for

TABLE I
ABLATION STUDY ON HYPERPARAMETERS α AND β IN (4) FOR Φ_R

α	β	Reconstruction loss	Final number of atoms	μ^+
0.8	1.4	9.07	6	0.19
0.85	1.4	9.86	5	0.20
0.9	1.4	10.59	4	0.21
1	1.5	11.37	3	0.10
1.7	1.5	13.56	2	0.23

unseen data and may lead the algorithm to become overfitted on the training set. As another perspective, trimming factors α and β iteratively reduce the degree-of-freedom of the algorithm and prevent the model from making redundant atoms. α and β can be empirically fine-tuned to reach the ideal number of minimally-overlapping atoms which is equal to the intrinsic dimensionality of embedding space (δ). Although the normal range of hyperparameters α and β heavily depends on the application, for the examples of this work the empirical range is $0.1 \leq \alpha,\ \ \beta \leq 2.5$.

The hyperparameter γ regulates the relative importance of cross-correlation versus reconstruction loss in dictionary temporal fine-tuning post-training stage. Since capturing exact temporal position of each floating atom for each test sample has higher priority than reconstruction loss, empirically observed that $\gamma = 0.5$ is an acceptable choice for our experiments.

C. Final Number of Active Atoms

Determining the value of intrinsic dimensionality of embedding space (δ) needs field knowledge and depends on the task, the target variable for the investigation (e.g., translational data of the suturing task along *x* axis), and the number of independent sub-tasks within the given structured task (e.g., number of gestures in surgical trial). A good heuristic for finding this number is to plot the manifold of reconstruction error and μ^+ versus the number of active atoms and wherever the reconstruction error and/or μ^+ do not reduce with increasing the number of active atoms (i.e., the elbow of the manifold) we assign that number of atoms to δ. For instance, in Table I $\delta = 3$ is an ideal final number of atom since $\delta \geq 4$ suffers from large overlap between atoms (i.e., high value of μ^+) and $\delta = 2$ suffers from both poor reconstruction loss and large overlap between atoms. This result also makes intuitive sense; Φ_R trajectory in suturing task is composed of three main gestures: passing the needle inside the tissue, pulling the needle from the tissue, and passing the needle from one hand to the other one.

D. Rotational Data vs. Translational Data

Our observations suggest that, rotational data offer more interpretation and insight about the quality of executive tasks with sharper distinction between data clusters in the embedding space. One possible reason that rotational data are more *expressive* than translational data is that rotational patterns are more closely related to the skills level of the user. This is based on the intuition that humans according to their advanced motor-control capabilities, can accomplish a lot of complicated tasks by performing a succession of several motions, but the quality, dexterity, and efficiency will be determined based on how well they perform rotations while executing translations. As an example, bipedal robots exhibit the same problem. Although they follow human joint trajectories in walking task, the overall behavior of their walking is different from that of humankind.

Another important reason is the *curse of extra details* in translational data, which masks general patterns with unnecessary microscale motions and prevents meaningful atom generation. The reason might be due to minor mistakes users unconsciously correct with simple motions rather than sophisticated rotations. This makes the subject's behavior more streamlined in the rotation space and noisier in translation space. Finally, compared to other types of surgery, such as eye surgery, minimally invasive surgery offers many translations, which can increase the effects of random motions as well.

E. Applications in Bi-Manual Tasks

In the proposed method, each trajectory component was investigated independently to analyze the performance of the executive task. However, in bimanual tasks (i.e., a class of tasks in which the brain must simultaneously plan and control the movements of both hands such as tying shoelaces or basic surgical tasks) what defines a person as an expert surgeon is not just simply what he/she performs by each individual hand but what he/she plans for the next step by executing a complex sequence of coordinated actions between his/her hands [33]. This concept is equivalent to the notion of *hands coordination* which measures the synchronicity and relationship between different trajectory components of one hand or between two hands. Incorporating coordination data together with the generated codes of the proposed method may benefit the quality of the final representation for downstream tasks such as skills classifier network.

VI. CONCLUSION

A new technique for visualizing surgical trajectories in the lowest possible dimensional space was presented in this paper. Representing trajectories based on a small number of minimally-overlapping atoms allowed us to meaningfully and intuitively decompose and investigate each trajectory. Each minimally-overlapping atom can be considered as a building block of the whole executive task that meaningfully reflects the style, skills, faults, and hidden behaviors of the user during performing the task. Incorporating floating atoms to capture important variations appearing at different temporal positions within the trajectory, improves model's accuracy and prevents information loss during the mapping procedure. All of these important features are objectivity expressed in terms of numerical gains in embedding (or code) space as an informative feature map for educational and examination purposes. According to our experiments on the JIGSAWS data set, our method is effective, reliable, and accurate for skills assessment and fault detection.

Appendix A
Proof of Property 1

According to *Hölder's inequality*, for any $\kappa, \upsilon \in (1, \infty)$ with $\frac{1}{\kappa} + \frac{1}{\upsilon} = 1$, we have

$$\sum_{k=1}^{N} |x_k y_k| \leq \left(\sum_{k=1}^{N} |x_k|^{\kappa}\right)^{\frac{1}{\kappa}} \left(\sum_{k=1}^{N} |y_k|^{\upsilon}\right)^{\frac{1}{\upsilon}} \tag{A.1}$$

for all $\boldsymbol{x}^\top = (x_1, \ldots, x_N)$, $\boldsymbol{y}^\top = (y_1, \ldots, y_N) \in \mathbb{R}^N$. Considering the special case $\kappa = \upsilon = 2$ in Hölder's inequality, (A.1) for non-zero vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ becomes

$$0 \leq |\boldsymbol{x}|^\top |\boldsymbol{y}| \leq \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2 \overset{\boldsymbol{x}\neq\boldsymbol{0}}{\underset{\boldsymbol{y}\neq\boldsymbol{0}}{\Longrightarrow}} 0 \leq \frac{|\boldsymbol{x}|^\top |\boldsymbol{y}|}{\|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2} \leq 1. \tag{A.2}$$

Applying (A.2) into (3) and considering the fact that $\mu(\boldsymbol{D}) \leq \mu^+(\boldsymbol{D})$ yields $0 \leq \mu(\boldsymbol{D}) \leq \mu^+(\boldsymbol{D}) \leq 1$ for any possible dictionary $\boldsymbol{D}$. ■

Appendix B
Convergence of Algorithm 1

Consider the convex optimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^N} \boldsymbol{f}(\boldsymbol{x}).$$

Definition 1 (η-Strongly Convex Function): The following statements are all equivalent to the condition that a differentiable function $\boldsymbol{f}$ is strongly convex with constant $\eta > 0$

1) $\boldsymbol{f}(\delta\boldsymbol{x} + (1-\delta)\boldsymbol{y}) \leq \delta\boldsymbol{f}(\boldsymbol{x}) + (1-\delta)\boldsymbol{f}(\boldsymbol{y}) - \frac{\delta(1-\delta)\eta}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$ for $\delta \in [0, 1]$.
2) The function $\tilde{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) - \frac{\eta}{2}\|\boldsymbol{x}\|_2^2$ is convex, $\forall \boldsymbol{x}$.

Definition 2: The convex optimization objective function $\boldsymbol{f}$ has component-wise L-Lipschitz continuous gradient if

$$|\nabla_i \boldsymbol{f}(\boldsymbol{x} + h\boldsymbol{e}_i) - \nabla_i \boldsymbol{f}(\boldsymbol{x})| \leq L|h| \tag{A.3}$$

where $\boldsymbol{x} \in \mathbb{R}^N$, $h \in \mathbb{R}$, $i = 1, \ldots, N$, and $\boldsymbol{e}_i$ is the standard basis vector of i^{th} component in $\boldsymbol{x}$.

Lemma 1: The criterion $\mathcal{L} = \sum_{i=1}^{n} \|\boldsymbol{x}_i - \sum_{j=1}^{p} \boldsymbol{d}_j \boldsymbol{c}_{ij}\|_2^2$ equals the quadratic function $(\boldsymbol{D}-\boldsymbol{D}^*)^\top \boldsymbol{C}^\top \boldsymbol{C}(\boldsymbol{D}-\boldsymbol{D}^*)$ plus a constant where $\boldsymbol{D}^*$ is the optimal solution of $\boldsymbol{D}$ in minimizing $\mathcal{L}$.

Proof: See [28]. ■

Lemma 2: The matrix $\boldsymbol{Q} = \boldsymbol{C}^\top\boldsymbol{C}$ defined in Lemma 1 is positive definite.

Proof: At first, we will prove that $\boldsymbol{Q}$ is a positive semidefinite matrix. According to the definition, the matrix $\boldsymbol{Q}$ is positive semidefinite matrix if $\boldsymbol{z}^\top\boldsymbol{Q}\boldsymbol{z} \geq 0$, $\forall \boldsymbol{z} \in \mathbb{R}^N$. We have

$$\boldsymbol{z}^\top\left(\boldsymbol{C}^\top\boldsymbol{C}\right)\boldsymbol{z} = (\boldsymbol{C}\boldsymbol{z})^\top(\boldsymbol{C}\boldsymbol{z}) = \|\boldsymbol{C}\boldsymbol{z}\|_2^2 \geq 0.$$

Now, to prove that $\boldsymbol{Q}$ is positive definite, we just need to prove that $\boldsymbol{z}^\top\boldsymbol{Q}\boldsymbol{z} \neq 0$, $\forall \boldsymbol{z} \neq \boldsymbol{0}$. Consider that $\boldsymbol{z}^\top\boldsymbol{Q}\boldsymbol{z} = 0$ for some $\boldsymbol{z} \neq \boldsymbol{0}$. It yields that $(\boldsymbol{D}-\boldsymbol{D}^*)^\top\boldsymbol{C}^\top\boldsymbol{C}(\boldsymbol{D}-\boldsymbol{D}^*)$ can be equal to zero for some $\boldsymbol{D} \neq \boldsymbol{D}^*$. It means, quadratic objective function $\mathcal{L}$ has more that one optimal solution. Contradiction. As a result, $\boldsymbol{Q} = \boldsymbol{C}^\top\boldsymbol{C}$ is positive definite and $\lambda_{\min}(\boldsymbol{Q}) > 0$ where $\lambda_{\min}(\boldsymbol{Q})$ is the smallest eigenvalue of $\boldsymbol{Q}$. ■

Lemma 3: The criterion $\mathcal{L}$ defined in Lemma 1 is η-strongly convex function with $\eta = 2\lambda_{\min}(\boldsymbol{C}^\top\boldsymbol{C}) > 0$.

Proof: According to Definition 1(ii), $\tilde{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) - \frac{\eta}{2}\|\boldsymbol{x}\|_2^2$ is convex if $\nabla^2\tilde{\boldsymbol{f}}(\boldsymbol{x}) = \nabla^2\boldsymbol{f}(\boldsymbol{x}) - \eta\boldsymbol{I} \succeq \boldsymbol{0}$ where $\boldsymbol{A} \succeq \boldsymbol{0}$ means that matrix $\boldsymbol{A}$ is positive semidefinite. As a result, a twice continuously differentiable $\boldsymbol{f}$ is η-strongly convex if $\nabla^2\boldsymbol{f}(\boldsymbol{x}) \succeq \eta\boldsymbol{I}$, $\forall\boldsymbol{x}$ or equivalently, the smallest eigenvalue of $\nabla^2\boldsymbol{f}(\boldsymbol{x})$ satisfies $\lambda_{\min}(\nabla^2\boldsymbol{f}(\boldsymbol{x})) \geq \eta$, $\forall\boldsymbol{x}$. According to the quadratic form of the criterion $\mathcal{L}$, $\nabla^2\boldsymbol{f}(\boldsymbol{x}) = \nabla^2\mathcal{L} = 2\boldsymbol{Q}$. As a result, $\mathcal{L}$ is η-strongly convex function with $\eta = 2\lambda_{\min}(\boldsymbol{C}^\top\boldsymbol{C})$ which according to Lemma 2, $\eta > 0$. ■

Lemma 4: Let $\boldsymbol{\mathcal{F}} = \boldsymbol{f} + \boldsymbol{g}$ where $\boldsymbol{f}$ is η-strongly convex function and $\boldsymbol{g}$ is a convex function (not necessarily strongly convex), then $\boldsymbol{\mathcal{F}}$ is η-strongly convex function.

Proof: According to Definition 1(i)

$$\begin{aligned}\boldsymbol{\mathcal{F}}(\delta\boldsymbol{x} + (1-\delta)\boldsymbol{y}) &= \boldsymbol{f}(\delta\boldsymbol{x} + (1-\delta)\boldsymbol{y}) + \boldsymbol{g}(\delta\boldsymbol{x} + (1-\delta)\boldsymbol{y}) \\ &\overset{*}{\leq} \delta\boldsymbol{f}(\boldsymbol{x}) + (1-\delta)\boldsymbol{f}(\boldsymbol{y}) - \frac{\delta(1-\delta)\eta}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \\ &\quad + \boldsymbol{g}(\delta\boldsymbol{x} + (1-\delta)\boldsymbol{y}) \\ &\overset{**}{\leq} \delta\boldsymbol{\mathcal{F}}(\boldsymbol{x}) + (1-\delta)\boldsymbol{\mathcal{F}}(\boldsymbol{y}) - \frac{\delta(1-\delta)\eta}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2\end{aligned}$$

where inequality * follows from the fact that $\boldsymbol{f}$ is strongly convex and inequality $**$ holds since $\boldsymbol{g}$ is convex. ■

Now, we want to prove the convergence of Algorithm 1 that tries to minimize (4). Both (5) and (10) are composed of a reconstruction error that according to Lemma 1 equals to a strongly convex quadratic function plus a convex l_1 normalization part. As a result, according to Lemma 4, (5) and (10) are strongly convex functions. In $\boldsymbol{C}$-step, we minimize (5) which according Lemma 3 is strongly convex with $\eta_1 = 2\lambda_{\min}(\boldsymbol{C}^\top\boldsymbol{C}) > 0$. In $\boldsymbol{D}$-step we minimize (10) which according Lemma 3 is strongly convex with $\eta_2 = 2\lambda_{\min}(\boldsymbol{D}\boldsymbol{D}^\top) > 0$. As a result, (4) is strongly convex with $\eta = \min\{\eta_1, \eta_2\} > 0$.

Lemma 5: The cost function (5) is directional component-wise Lipschitz continuous gradient with $L_1 = \max_i\{\|\boldsymbol{d}_i\|_2^2\}$ in $\boldsymbol{C}$-step domain.

Proof: The gradient of (5) with respect to a parameter c_{ij} (i.e., the j^{th} component of a code vector $\boldsymbol{c}_i$) is

$$\nabla_{c_{ij}}\mathcal{L}_C^+ = \boldsymbol{d}_j^\top(\boldsymbol{D}\boldsymbol{c}_i - \boldsymbol{x}_i) + \alpha\ \operatorname{sign}\big(c_{ij}\big). \tag{A.4}$$

Equation (A.4) is also used as a part of update rule in `sparseCode` algorithm to calculate the optimal value of code matrix $\boldsymbol{C}$. Due to the limitations rising from the learning rate and approximation nature of `sparseCode` algorithm, in cases that $c_{ij} \equiv 0$ the algorithm fails to land on $c_{ij} = 0$ and we have a fluctuation around the origin since $\operatorname{sign}(c_{ij})$ fluctuates between -1 and 1. A common heuristic applied in this situation is to *clamp* c_{ij} to zero (i.e., manually setting c_{ij} to zero) to prevent such fluctuations. Under these circumstances, we can assume that $\operatorname{sign}(c_{ij})$ does not change and is the same for $\nabla_i\boldsymbol{f}(\boldsymbol{x}+h\boldsymbol{e}_i)$ and $\nabla_i\boldsymbol{f}(\boldsymbol{x})$ in Definition 2. This assumption leads us to the directional component-wise L-Lipschitz continuous gradient in $\boldsymbol{C}$-step domain. According to (A.3) we should have

$$\left|\boldsymbol{d}_j^\top\boldsymbol{d}_j h e_{ij}\right| = \left|\boldsymbol{d}_j^\top\boldsymbol{d}_j\right||h| = \left\|\boldsymbol{d}_j\right\|_2^2\ |h| \leq L|h|. \tag{A.5}$$

We can reach to (A.5) for all components of code matrix $\boldsymbol{C}$. As a result, a good choice for L is the length of largest code vector $\boldsymbol{d}_j$ in the dictionary matrix $\boldsymbol{D}$, or equivalently, $L_1 = \max_i\{\|\boldsymbol{d}_i\|_2^2\}$. ■

Corollary 1: According to Lemma 5, the cost function (10) is directional component-wise Lipschitz continuous gradient with $L_2 = \max_j\{\|\boldsymbol{c}_j\|_2^2\}$ in $\boldsymbol{D}$-step domain.

Corollary 2: According to Lemma 5 and Corollary 1, the cost function (4) is directional component-wise Lipschitz continuous gradient with $L = \max\{L_1,\ L_2\}$ in the dual-sparse coding algorithm domain.

According to [29], [30] and the values of η and L calculated above, for each iteration $k \geq 0$ for Algorithm 1 starting from $\boldsymbol{C}_0$ and $\boldsymbol{D}_0$, we have

$$\begin{aligned}&\mathbb{E}\big[\mathcal{L}^+(\boldsymbol{D}_{k+1}, \boldsymbol{C}_{k+1})\big] - \mathcal{L}^+\big(\boldsymbol{D}^*, \boldsymbol{C}^*\big)\\ &\quad\leq \left(1 - \frac{\eta}{2Ldp^2n}\right)\big[\mathcal{L}^+(\boldsymbol{D}_k, \boldsymbol{C}_k) - \mathcal{L}^+\big(\boldsymbol{D}^*, \boldsymbol{C}^*\big)\big] \quad \text{(A.6)}\end{aligned}$$

where $\boldsymbol{D}^*$ and $\boldsymbol{C}^*$ are local optima of (4) and d, p, and n are dimensions of matrices $\boldsymbol{D}$ and $\boldsymbol{C}$ defined in (1). Note that, if in Lemma 5, our cost function was component-wise Lipschitz continuous gradient, not *directional* component-wise Lipschitz continuous gradient, $\boldsymbol{D}^*$ and $\boldsymbol{C}^*$ are global optimum solutions. $2dp^2n$ is the total number of components in $\boldsymbol{D}$ and $\boldsymbol{C}$ which is the total number of parameters for the optimization problem (4). Although the values of η and L depend on the matrices $\boldsymbol{D}$ and $\boldsymbol{C}$ in the previous step, the convergence of expected error to zero (i.e., reaching to local or global optima) in each iteration is guaranteed in (A.6) since the value of $1 - \eta/(2Ldp^2n)$ is always less than 1.

References

[1] A. Soleymani, X. Li, and M. Tavakoli, "Deep neural skill assessment and transfer: Application to robotic surgery training," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021, pp. 8822–8829.

[2] A. Soleymani, A. A. S. Asl, M. Yeganejou, S. Dick, M. Tavakoli, and X. Li, "Surgical skill evaluation from robot-assisted surgery recordings," in *Proc. IEEE Int. Symp. Med. Robot. (ISMR)*, 2021, pp. 1–6.

[3] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Accurate and interpretable evaluation of surgical skills from kinematic data using fully convolutional neural networks," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 9, pp. 1611–1617, 2019.

[4] I. Funke, S. T. Mees, J. Weitz, and S. Speidel, "Video-based surgical skill assessment using 3D convolutional neural networks," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 7, pp. 1217–1225, 2019.

[5] H. Doughty, W. Mayol-Cuevas, and D. Damen, "The pros and cons: Rank-aware temporal attention for skill determination in long videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7862–7871.

[6] N. Sünderhauf *et al.*, "The limits and potentials of deep learning for robotics," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, pp. 405–420, 2018.

[7] B. van Amsterdam, M. Clarkson, and D. Stoyanov, "Gesture recognition in robotic surgery: A review," *IEEE Trans. Biomed. Eng.*, vol. 68, no. 6, pp. 2021–2035, Jun. 2021.

[8] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 47–54.

[9] G. Menegozzo, D. Dall'Alba, C. Zandonà, and P. Fiorini, "Surgical gesture recognition with time delay neural network based on kinematic data," in *Proc. IEEE Int. Symp. Med. Robot. (ISMR)*, 2019, pp. 1–7.

[10] R. DiPietro *et al.*, "Recognizing surgical activities with recurrent neural networks," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent.*, 2016, pp. 551–558.

[11] R. DiPietro *et al.*, "Segmenting and classifying activities in robot-assisted surgery with recurrent neural networks," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 14, no. 11, pp. 2005–2020, 2019.

[12] D. Itzkovich, Y. Sharon, A. Jarc, Y. Refaely, and I. Nisky, "Using augmentation to improve the robustness to rotation of deep learning segmentation in robotic-assisted surgical data," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 5068–5075.

[13] B. van Amsterdam, M. J. Clarkson, and D. Stoyanov, "Multi-task recurrent neural network for surgical gesture recognition and progress prediction," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 1380–1386.

[14] D. Liu and T. Jiang, "Deep reinforcement learning for surgical gesture segmentation and classification," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Intervent.*, 2018, pp. 247–255.

[15] J. Rosen, M. Solazzo, B. Hannaford, and M. Sinanan, "Task decomposition of laparoscopic surgery for objective evaluation of surgical residents' learning curve using hidden markov model," *Comput.-Aided Surg.*, vol. 7, no. 1, pp. 49–61, 2002.

[16] A. Derathé, F. Reche, P. Jannin, A. Moreau-Gaudry, B. Gibaud, and S. Voros, "Explaining a model predicting quality of surgical practice: A first presentation to and review by clinical experts," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 16, no. 11, pp. 2009–2019, 2021.

[17] I. Tošić and P. Frossard, "Dictionary learning," *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

[18] A. L. Chistov and D. Y. Grigor'Ev, "Complexity of quantifier elimination in the theory of algebraically closed fields," in *Proc. Int. Symp. Math. Found. Comput. Sci.*, 1984, pp. 17–31.

[19] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, no. 1, pp. 19–60, 2010.

[20] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Stat.*, vol. 32, no. 2, pp. 407–499, 2004.

[21] C. Bao, H. Ji, Y. Quan, and Z. Shen, "Dictionary learning for sparse coding: Algorithms and convergence analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 7, pp. 1356–1369, Jul. 2016.

[22] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[23] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.

[24] C. Bao, J.-F. Cai, and H. Ji, "Fast sparsity-based orthogonal dictionary learning for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 3384–3391.

[25] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[26] C. Chen *et al.*, "Crowd-sourced assessment of technical skills: A novel method to evaluate surgical performance," *J. Surg. Res.*, vol. 187, no. 1, pp. 65–71, 2014.

[27] Y. Gao *et al.*, "JHU-ISI gesture and skill assessment working set (JIGSAWs): A surgical activity dataset for human motion modeling," in *Proc. Miccai Workshop M2CAI*, vol. 3, 2014, p. 3.

[28] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc. B Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

[29] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, 2012.

[30] P. Richtárik and M. Takáč, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Math. Program.*, vol. 144, no. 1, pp. 1–38, 2014.

[31] T. N. Judkins, D. Oleynikov, and N. Stergiou, "Objective evaluation of expert and novice performance during robotic surgical training tasks," *Surg. Endoscopy*, vol. 23, no. 3, pp. 590–597, 2009.

[32] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 3501–3508.

[33] S. S. Vedula, A. Malpani, N. Ahmidi, S. Khudanpur, G. Hager, and C. C. G. Chen, "Task-level vs. segment-level quantitative metrics for surgical skill assessment," *J. Surg. Educ.*, vol. 73, no. 3, pp. 482–489, 2016.