Unsupervised identification of surgical robotic actions from small non-homogeneous datasets

Daniele Meli, Paolo Fiorini

***Abstract*—Robot-assisted surgery is an established clinical practice. The automatic identification of surgical actions is needed for a range of applications, including performance assessment of trainees and surgical process modeling for autonomous execution and monitoring. However, supervised action identification is not feasible, due to the burden of manually annotating recordings of potentially complex and long surgical executions. Moreover, often few example executions of a surgical procedure can be recorded. This paper proposes a novel fast algorithm for unsupervised identification of surgical actions in a standard surgical training task, the ring transfer, executed with da Vinci Research Kit. Exploiting kinematic and semantic visual features automatically extracted from a very limited dataset of executions, we are able to significantly outperform state-of-the-art results on a dataset of non-expert executions (58% vs. 24% F1-score), and improve performance in the presence of noise, short actions and non-homogeneous workflows, i.e. non repetitive action sequences.**

***Index Terms*—Robotic surgery, semantic visual features, unsupervised gesture recognition**

I. Introduction

ROBOT assisted minimally invasive surgery (RAMIS) has improved the quality of standard laparoscopic surgery (precise positioning and tremor filtration, visual immersion, faster recovery time for the patient) in a number of clinical scenarios, including urology [1], esophagectomy [2] and pancreatectomy [3]. Surgical robots as the well established da Vinci® from Intuitive Surgical are currently employed by novice surgeons for training. Automatic quality assessment of trainees by a supervisory system is needed to improve the error-prone verification made by teaching experts. A supervisory system will be needed also in the *operating room of the future* [4], to improve the safety and efficiency of surgery and assist medical staff in decision making. Moreover, a major challenge for the operating room of the future will be the implementation of autonomous robotic surgery to reduce hospital costs and surgeon fatigue. This problem has been only partially addressed, mainly using pre-defined finite state models [5], [6], statistical models [7] and logics [8], [9]. All the mentioned issues require the definition of an accurate *surgical process model* (SPM) describing the surgical procedure. The definition of the SPM depends on the granularity level of analysis (from low-level description of elementary motory *actions* to high-level sequence of main phases of an intervention) [10]. Though it is possible to define a prior standard SPM from available expert surgical knowledge, learning from surgical practice is needed to build a robust SPM which properly describes variability of the patient's anatomy and the surgical workflow, especially at the action level. Datasets of RAMIS usually include kinematic records from the robot and videos of the execution. Given expert annotations of the dataset describing the workflow of execution, Bayes models [11] and deep neural networks [12], [13] can be successfully used for automatic supervised learning of SPMs. However, manual annotation of extensive surgical datasets is prone to errors and tedious. Current research is mostly focused on the unsupervised identification of actions from surgical datasets. Inspired from advances in human activity recognition [14], [15], researchers have investigated approaches to unsupervised recognition of surgical actions. [16], [17] use transition-state clustering (TSC) with deep neural networks on videos and kinematics of JIGSAWS dataset of surgical actions [18]. Approaches based on statistical models, e.g. TSC with Gaussian mixture models [19], soft-boundary algorithms with fuzzy clustering scores [20] and weakly supervised algorithms [21] improve the accuracy. However, they use extensive datasets as JIGSAWS, containing many *homogeneous* repetitions of the same task, i.e. executions which do not differ in the sequence of actions, but present only kinematic variations.

In this paper, we propose a novel algorithm for unsupervised surgical action identification, considering the benchmark training task of ring transfer with da Vinci Research Kit (dVRK) [22]. The setup for the task is shown in Figure 1. The goal of the task is placing rings on the same-colored pegs, using the two patient-side manipulators (PSMs) of dVRK. Dynamic geometric conditions on the setup affect the sequence of actions. Rings may either be grasped and placed by the same arm or be transferred between arms, for economy of motion. Pegs can be occupied by other rings, so they must be freed before placing another ring. Moreover, rings may be on pegs, thus requiring extraction, or on the base.

Differently from most state-of-the-art algorithms, this paper aims at overcoming some non-addressed limitations of unsupervised surgical action identification, specifically: 1) dealing with small surgical datasets; 2) recognizing actions with short duration; and 3) dealing with non-homogeneous datasets of executions, i.e. differing in the operative conditions and the flow of actions (emulating anatomy-dependent variability in surgery). To the best of our knowledge, the problem of non-homogeneous datasets has been only partially addressed recently in [23]. Authors propose a method based on deep

Manuscript received: May 16, 2021; Revised: June 30, 2021; Accepted: August 5, 2021.

This paper was recommended for publication by Editor Jessica Burgner-Kahrs upon evaluation of the Associate Editor and Reviewers' comments.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No. 742671 (ARS).

Authors are with Dept. of Computer Science, University of Verona, Italy. Contact author: Daniele Meli, daniele.meli@univr.it

Digital Object Identifier (DOI): see top of this page.

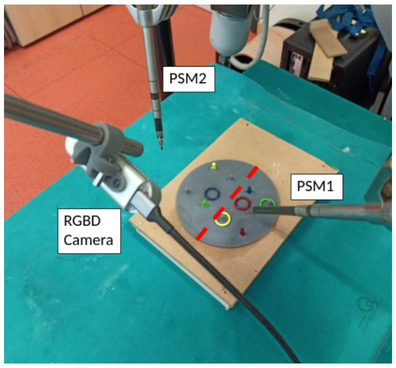


Fig. 1: The setup for the ring transfer task. The red dashed line defines reachability regions for the two PSMs.

learning from augmented simulated data, and apply it to a peg transfer task similar to ours. The problem of identifying actions in small datasets of peg transfer has been investigated in [24].

Inspired from the advances in semantic video interpretation from high-level knowledge [25], our algorithm enriches purely kinematic features with semantic domain-dependent features automatically computed from videos of execution.

The rest of the paper is organized as follows: in Section II we describe our algorithm; in Section III we compare the performance of our algorithm, mainly in comparison with [24]; finally, Section IV concludes the paper, in comparison and outlines possible future research.

II. Action identification algorithm

The input to our action identification algorithm is an *execution trace*, i.e. the kinematic and video stream of an instance of the task; the output is a set of *clustered segments*, i.e. parts of the execution trace corresponding to the same action. Ring transfer consists of 6 actions: `move(A,O,C)` which defines motion of one PSM `A` to an object `O` $\in$ `{ring,peg}` with color `C`; `move(A,center,C)` which defines the motion of one PSM to the center of the base to transfer a ring to the other arm; `grasp(A,ring,C)`, `release(A)` which define the grasping and releasing action of a colored ring, respectively; `extract(A,ring,C)` which defines the extraction of one ring from a peg where it is placed.

Our algorithm consists of 2 steps: *action segmentation* to identify changepoints delimiting segments in the execution trace; and *action classification* to group together segments corresponding to the same action class.

A. Execution trace

The execution trace of the task consists of a kinematic signature and a synchronized semantic video stream.

1) Kinematic signature: It contains relevant kinematic features to describe the trace of execution. Similarly to [24] chosen as benchmark for this paper, our kinematic signature consists of 16 quantities, including the Cartesian $\mathbf{p}_A = \{x_{pos,A}, y_{pos,A}, z_{pos,A}\}$ and the quaternion coordinates $\mathbf{q}_A = \{x_{or,A}, y_{or,A}, z_{or,A}, w_{or,A}\}$ describing the position and orientation of the end effectors of the arms `A` of the dVRK robot; and the opening angles j_A of the grippers. Additional kinematic features may be considered, e.g., speed of the end effectors as in [21] and scale-invariant measures of torsion and curvature of the Cartesian trajectories as in [24]. After preliminary testing with all the features, we decide to omit velocities, curvature and torsion since they do not affect our results.

TABLE I: Geometric features detected in the frames of the video stream of the execution traces by the SA algorithm.

Name	Description
$\mathbf{p}_{rc} = \{x_{rc}, y_{rc}, z_{rc}\}$	position of ring with color `C` (center)
$\mathbf{p}_{pc} = \{x_{pc}, y_{pc}, z_{pc}\}$	position of peg with color `C` (tip)
$\mathbf{p}_b = \{x_b, y_b, z_b\}$	position of center of peg base
rr	ring radius

2) Semantic video stream: It consists of a video stream of the execution trace acquired from a RGB-D camera (calibrated with PSMs as in [26]). Geometric features described in Table I are extracted from video frames using standard color segmentation and shape recognition (to distinguish between pegs and rings) with Random Sample Consensus [27]. Then, they are combined with kinematic features and translated to semantic features called *fluents* with the situation awareness (SA) algorithm proposed in [9], according to the following relations:

$$\begin{aligned}
\texttt{at(A,ring,C)} &\leftarrow ||\mathbf{p}_A - \mathbf{p}_{rc}||_2 < rr \\
\texttt{at(A,peg,C)} &\leftarrow ||\mathbf{p}_A - \mathbf{p}_{pc}||_2 < rr \ \wedge \\
&\wedge \ z_{pc} < z_{pos,A} \\
\texttt{in\_hand(A,ring,C)} &\leftarrow ||\mathbf{p}_A - \mathbf{p}_{rc}||_2 < rr \ \wedge \ j_A < \frac{\pi}{8}[1] \\
\texttt{on(ring,C1,peg,C2)} &\leftarrow ||\mathbf{p}_{rc1} - \mathbf{p}_{pc2}||_2 < rr \ \wedge \\
&\wedge \ z_{rc1} < z_{pc2} \\
\texttt{reachable(A}_x\texttt{,O,C)} &\leftarrow argmin_A |y_{oc} - y_{pos,A}| = A_x \\
\texttt{closed\_gripper(A)} &\leftarrow j_A < \frac{\pi}{8} \\
\texttt{at(A,center)} &\leftarrow ||\{x_{pos,A}, y_{pos,A}\} - \{x_b, y_b\}||_2 < \\
&< rr
\end{aligned}$$

The semantics of fluents is clear. In particular, `reachable(A,O,C)` defines which objects are in the reachability region of each PSM, as delimited by the red dashed line in Figure 1.

B. Action segmentation

Given an execution trace, the segmentation algorithm 1 identifies changepoints corresponding to starting / ending timesteps of actions of the task. The algorithm involves two main steps: changepoint detection from kinematics, and changepoint filtering from the semantic video stream.

1) Changepoint detection: The features in the kinematic signature are first normalized by their maximum values to allow comparison between different configurations of the setup for the task (different locations of objects in the scene and

[1] $\frac{\pi}{8}$ is chosen empirically to identify that the gripper is closed, since the angle is not 0 when the PSM is holding a ring.

distances to the PSMs). Then, filtering at $1.5\,\mathrm{Hz}$ is applied to each feature, to consider only relevant motions within the fundamental frequency of human gestures [28]. We identify changepoints in the kinematic signature evaluating peaks in the 2nd derivative of the features, and only peaks above a percentage α of the maximum absolute value are considered. In order to remove implicit noise in the 2nd derivative, we compute it using Savitzky-Golay filter (SGF) [29], a digital filter based on polynomial fitting over an adjustable time window. SGF captures higher-order momenta in the data, while removing noise in the signal. SGF has a long history of successful applications in time series analysis, e.g. for kinematic analysis of the human arm [30] and neural signals [31]. Notice that there exist several algorithms and implementations for changepoint detection, and an extensive review can be found in [32]. We choose peak detection from SGF on the 2nd derivative of kinematic features because it is fast and effective for our experiments, and it is robust since it requires no prior assumption on the number of segments, and few parameters to be tuned (time window set to 21 timesteps and $\alpha = 20\%$ empirically).

2) Changepoint filtering: Changepoint detection may return spurious changepoints, corresponding e.g. to abrupt motions during the execution. Then, we reduce the set of changepoints from the detection algorithm, excluding consecutive changepoints distant less than $1s$ (average duration of shortest actions, see Section III-A). Then, we compute fluents from the video stream in correspondence of each changepoint, and we omit consecutive changepoints which share the same set of fluents. This choice is made under the assumption that *different actions derive from different environmental conditions*: hence, two consecutive changepoints defining the start of two different actions shall not share the same set of fluents. Our approach requires tuning of fewer parameters than the one proposed in [24]. In fact, authors of [24] propose to identify changepoints using persistence analysis [33]. Persistence analysis removes noise and identifies relevant local minima and maxima in the kinematic features, but requires the definition of an ad-hoc threshold for each feature, which is often task- and operator-dependent.

Algorithm 1 Action segmentation algorithm

```
1: Input: Execution trace with temporal kinematic signature $K(t)$ and video stream $V(t)$, threshold for peak detection $\alpha$
2: Output: Set of changepoints $C$
3: Initialize: Normalize and filter $K(t)$, $C = []$
4: % Changepoint detection
5: for $k(t)$ kinematic feature $\in K(t)$ do
6:     $\ddot{k}(t)$ = SGF($k(t)$)
7:     peaks = PeakDetect($\ddot{k}(t)$)
8:     for $p \in$ peaks do
9:         if $|\ddot{k}(p)| > \alpha \cdot \max|\ddot{k}(p)|$ then
10:            $C$.append($p$)
11: % Changepoint filtering
12: fluents $F = []$
13: $c_{old} = 0$
14: for $c \in C$ do
15:     old_fluents $F_{old} = F$
16:     $F$ = FluentCompute($V(c)$)
17:     if $F == F_{old} \vee c - c_{old} < 1s$ then
18:         $C$.remove($c$)
19:     $c_{old} = c$
20: return $C$
```

C. Action classification

Given the changepoints of the execution trace(s), we use k-NN classification to group segments corresponding to same actions. k-NN is preferred to other classification methods as support vector machines [24] and self-organizing maps [15] because it usually performs better on small datasets. We associate each segment in the execution trace with a feature vector $\mathbf{f} = [\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3]$, containing both kinematic information as in [24] and semantic environmental information from the video stream.

1) $\mathbf{f}_1$*:* It is an array of reals representing the kinematic features of each segment. Each feature k_i in the kinematic signature of the segment is shifted to start from $t = 0$ (for fair comparison between different segments of the same action); then, it is approximated by a polynomial $p_i(t) = \sum_{j=1}^{n} a_{j,i} t^j$, with $n \in \mathbb{N}$ arbitrary degree as in [24] ($n = 5$ empirically in this work). Then, $\mathbf{f}_1$ is built concatenating coefficients $a_{j,i}$, resulting in an array of dimension $(n+1) \cdot 16$. Polynomial approximation is not the only option for kinematic representation. For instance, in [15] Fourier coefficients are used for noise-robust action identification in the CAVIAR dataset for human gestures [34]. However, in our experiments Fourier approximation does not show to improve the performance, so it is discarded. We want our classification algorithm to generalize over the actual PSM executing a specific action: for instance, two segments corresponding to `move(PSM1,ring,red)`, `move(PSM2,ring,yellow)` must be classified in the same cluster, corresponding to the abstract action `move(A,ring,C)`. Hence, we check whether one of the PSMs does not move, i.e. the kinematic signatures at the starting and ending changepoints have no significant difference. In case only one arm moves, the coefficients of the polynomial approximations of the kinematic features for that specific arm are added to $\mathbf{f}_1$, and null coefficients for the other arm are appended to complete the vector. In case both PSMs are moving, coefficients of PSM1 and PSM2 are concatenated to build $\mathbf{f}_1$. In this way, there is no distinction whether an action is executed by one arm or the other.

2) $\mathbf{f}_2$*:* It is a Boolean array representing fluents holding at the beginning changepoint of each segment. Each entry in the array corresponds to a fluent defined in Section II-A, excluding `reachable` which is identified at all timesteps in the video stream (all rings and pegs are always reachable at least by one arm), hence it is neglected. Each entry in the array has true value if the corresponding fluent holds at the beginning of the segment, otherwise it is set to false. In order to guarantee the invariance of the classification algorithm with respect to the arm and color instances, $\mathbf{f}_2$ has two entries for each fluent, one per PSM, and the color attribute is ignored. Similarly to the generation of $\mathbf{f}_1$, if a fluent holds only for one arm, then the first corresponding entry is set to true, regardless of the specific arm. The final dimension of $\mathbf{f}_2$ is 12.

3) $\mathbf{f}_3$*:* It is a 16D Boolean vector with entries corresponding to each feature in the kinematic signature of the segment. If one signature varies from the beginning to the ending changepoint, the value of the corresponding entry is set to true (as for $\mathbf{f}_1$, the order of entries for the two arms does not depend on the specific PSM). $\mathbf{f}_3$ is needed because instances of the same action may significantly differ from a kinematic perspective, due to the different relative position of objects and arms in the scene (for instance, in the scenario depicted in Figure 1, `move(PSM1,ring,red)` requires a completely different motion than the one needed for `move(PSM2,ring,blue)`). However, we expect that the same kinematic features vary along the segment.

4) Classification algorithm: k-NN classification compares feature vectors for different segments in the execution trace(s) containing both Boolean and real values. Hence, we need to define a mixed distance metric $d_{i,l}$ between segments i, l:

$$d_{i,l} = \sqrt{\frac{d_e^2}{d_{emax}^2} + \frac{d_h^2}{d_{hmax}^2}} \tag{1}$$

where d_e is the standard Euclidean distance between coefficients in $\mathbf{f}_1$ for the two segments, $d_e = \sqrt{\sum_{j=1}^{n}(a_{j,i} - a_{j,l})^2}$; while d_h is the Hamming distance between $[\mathbf{f}_2, \mathbf{f}_3]$ for the two segments

$$d_h = \frac{h_{i,l}}{\dim([\mathbf{f}_2, \mathbf{f}_3]_i)}$$

being $h_{i,l}$ the number of different values in $[\mathbf{f}_2, \mathbf{f}_3]$ for segment i with respect to segment l, and $\dim(\cdot)$ the dimension of an array. d_{emax}, d_{hmax} are normalizing factors which are needed for fair comparison of Euclidean and Hamming distances, and they are chosen as the maximum cost resulting from k-NN classification with only Euclidean distance over $\mathbf{f}_1$ and only Hamming distance over $[\mathbf{f}_2, \mathbf{f}_3]$, respectively. k for k-NN classification is chosen as the number of occurrencies of the most frequent action in the dataset of executions. In this way, the algorithm is able to recognize all instances of actions in the dataset. A higher k value would also be valid; however, in the experiments we show that the best classification performance is achieved with the minimum k.

III. Experiments

We prove the advantages of our unsupervised action identification algorithm with three different experiments. In Test A, we consider nine executions of the ring transfer task in homogeneous *standard environmental condition*, i.e. with all rings placed on grey pegs and requiring transfer between arms. The task is executed by 3 users (not surgeons) with different expertise in using the dVRK. This experiment is useful for comparison with benchmark [24], which considers multiple homogeneous executions of the same task by three expert users.

In Test B, we first consider an execution in standard environmental conditions, performed autonomously as explained in [9], where the robot's trajectories were described as Dynamic Movement Primitives (DMPs) [35], [36], [37], [38]. DMPs allow to learn and replicate the shape of trajectories executed by humans, both in Cartesian and orientation space. In [9], 15 executions of the ring transfer task were recorded from the same users of Test A and used for learning trajectories of each action in the task. Hence, though the execution is autonomous, it reproduces the kinematics of human executions. In this test, we generate synthetic task replications from the original autonomous execution, adding low-frequency human-like kinematic noise to test the robustness of our algorithm with respect to noise (hence, the dataset is still homogeneous).

Finally, in Test C, we consider a non-homogeneous dataset consisting of only the standard execution of Test B without noise, and three autonomous executions of the task in unconventional environmental conditions, shown in Figure 3. All execution traces (kinematic readings from dVRK[2] and point clouds from a Realsense D435 RGB-D camera[3]) are collected with the Robot Operating System to ensure synchronization.

For a qualitative understanding especially of our changepoint filtering algorithm, Figure 2 shows the results of our segmentation algorithm applied to the execution depicted in Figure 3b. Our changepoint detection algorithm finds more changepoints (red solid lines) than real ones from manual segmentation (blue lines). However, we are able to filter out wrong changepoints (red dashed lines) exploiting fluent detection from video stream, when two consecutive changepoints share the same set of fluents.

We quantify the performance of our algorithm using the reference metrics proposed in [24]. Specifically, we rate the segmentation accuracy using the *matching score*:

$$M_i = \frac{|\cap(t_i, g_i)|}{|g_i|}$$

where t_i is the segment identified by our algorithm, g_i is the real segment corresponding to the i-th action, and $|\cdot|$ denotes the temporal length of a segment. The matching score measures the overlapping between the identified and actual segment, normalized by the length of the actual segment. The results of action classification are quantified using *precision, recall* and *F1-score*. Precision for an action class A is defined as:

$$Pr = \frac{TP}{FP + TP}$$

being TP the number of true positives, i.e. segments correctly classified in A, and FP the number of false positives, i.e. segments mistakenly classified in A. Precision measures the rate of success of the classification algorithm on the full dataset of executions. Since k for k-NN classification is chosen as the maximum number of occurrencies of the most frequent action in the dataset, $FP + TP$ is chosen for the i-th action class as $\dim(P)$, where P is the maximum set of segments such that:

- $\dim(P) \geq n_{occ,i}$, where $n_{occ,i}$ is the number of occurrencies of the i-th action in the dataset;
- the score of the last segment in P is $s_P = s_{max}$, being s_{max} the score of the $n_{occ,i}$-th segment in P.

[2]https://github.com/jhu-dvrk/sawIntuitiveResearchKit

[3]https://www.intelrealsense.com/depth-camera-d435/

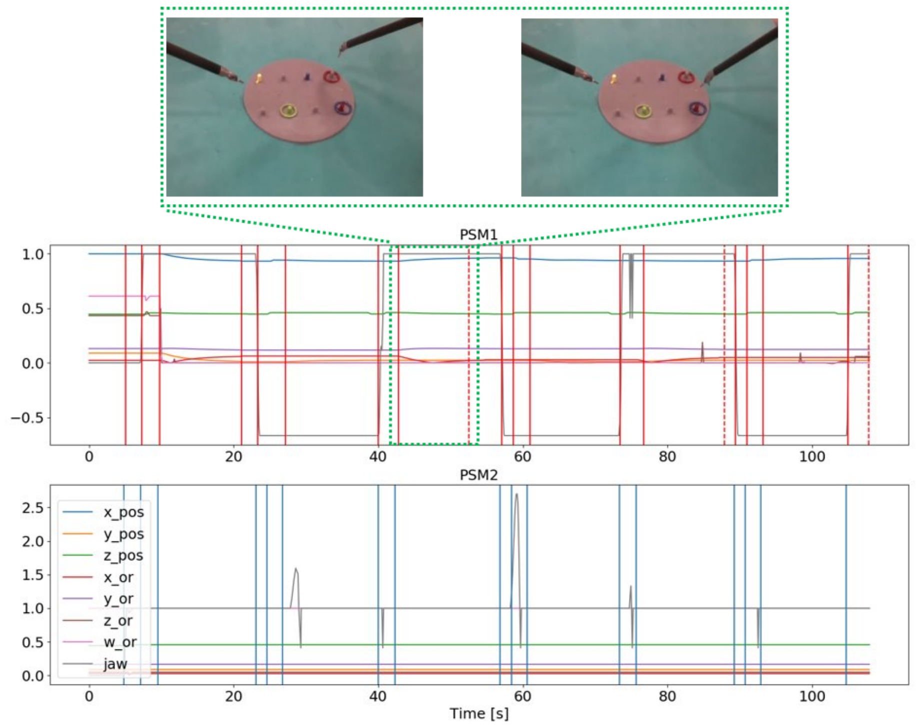


Fig. 2: Kinematic signature (after filtering) and segmentation results for the execution in Figure 3b. Blue vertical lines represent real changepoints, red solid lines represent changepoints identified by our algorithm, red dashed lines represent changepoints excluded after fluent detection. Frames on the top correspond to two consecutive changepoints sharing the same set of fluents (`on(ring,red,peg,grey), on(ring,blue,peg,red)` and the reachability fluents), so the second one is omitted.

In this way, the scores for less frequent actions are not significantly affected by the most frequent ones. Recall for an action class A is defined as:

$$Rec = \frac{TP}{FN + TP}$$

with FN the number of false negatives, i.e. segments mistakenly not included in A. Recall measures the rate of success of the classification algorithm for a single action. F1-score combines precision and recall as follows:

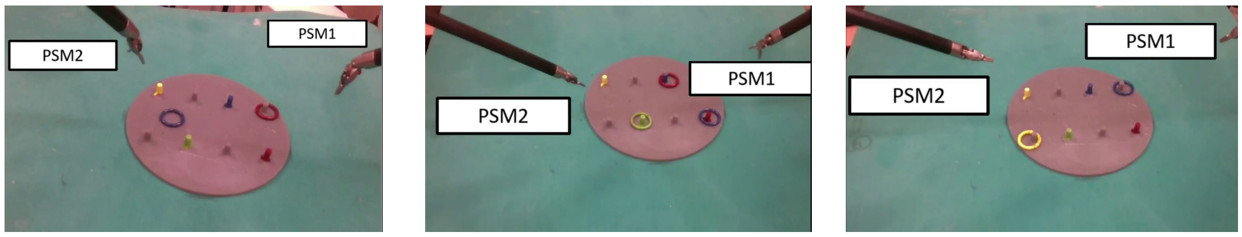
$$F1 = 2\frac{Pr \cdot Rec}{Pr + Rec}$$

A. Test A: Homogeneous dataset from human executions

In this test, we consider 9 executions of the ring transfer task in standard environmental conditions, executed by 3 users with different expertise in using the dVRK. Gripper actions and `extract` last 1.05 ± 1.09 s, while `move` actions last 3.69 ± 2.90 s (comparably with peg transfer in [24] on average). The high variance is due to the different expertise of users. Each execution trace contains 36 actions (actions appear multiple times in each execution). Table II reports the matching scores (for each action and on average over all actions). The segmentation results are slightly better than [24], though comparable. In Table III, we report the results of our classification algorithm. We choose $k = 36$ for k-NN classification, i.e. the number of occurrencies of the most frequent action (`release`) in the dataset. As explained in Section II-C, this is the minimum possible value for k to recognize at least all action instances. However, there is no prior guarantee that this is the optimal value for the classification performance. Hence, in Figure 4 we run a preliminary test wih increasing values of $k \in [36, 56]$ with a step of 2, and we compare the average F1-score over all actions in the dataset to identify the best choice of k. Results show that $k = 36$ is the optimal value. Hence, also in the following experiments we will choose k as the minimum possible value. Given k, we first use the full feature array $[\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3]$ presented in Section II-C as input to the classification algorithm. However, this results in poor performance, especially for short actions. Hence, we use only Boolean features $[\mathbf{f}_2, \mathbf{f}_3]$ for short actions to obtain the results in Table III. Results are significantly improved with respect to [24], whose method suffers from the executions by non-expert users, on the contrary.

TABLE II: Matching scores for Test A. All values are percentages.

Action		**Matching score**
`move(A,ring,C)`		85.65
`move(A,peg,C)`		90.69
`move(A,center,C)`		82.43
`grasp(A,ring,C)`		77.05
`extract(A,ring,C)`		82.73
`release(A)`		90.18
Average	Our method	**84.79**
	Method in [24]	81.90



(a) Execution with failure (fallen ring) (b) Execution with occupied pegs (c) Simultaneous execution by two PSMs

Fig. 3: Initial environmental conditions for the task executions used in Test B.

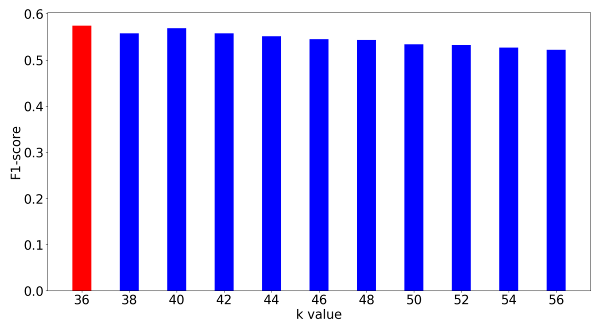


Fig. 4: Average F1-score over all actions for Test A, with increasing k value. The minimum value (red bar) is the optimal one.

TABLE III: Action classification results for Test A. All values are percentages.

Action	Feature array	Pr	Rec	$F1$
`move(A,ring,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	**100.00**	**83.33**	**90.91**
	$\mathbf{f_1}$ as in [24]	22.22	22.22	22.22
`move(A,peg,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	**55.56**	**55.56**	**55.56**
	$\mathbf{f_1}$ as in [24]	19.44	19.44	19.44
`move(A,center,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	**58.33**	**58.33**	**58.33**
	$\mathbf{f_1}$ as in [24]	25.00	25.00	25.00
`grasp(A,ring,C)`	$[\mathbf{f_2}, \mathbf{f_3}]$	**41.23**	**40.00**	**40.61**
	$\mathbf{f_1}$ as in [24]	17.78	22.22	19.66
`extract(A,ring,C)`	$[\mathbf{f_2}, \mathbf{f_3}]$	**66.67**	**66.67**	**66.67**
	$\mathbf{f_1}$ as in [24]	30.56	30.56	30.56
`release(A)`	$[\mathbf{f_2}, \mathbf{f_3}]$	**40.00**	**40.00**	**40.00**
	$\mathbf{f_1}$ as in [24]	26.67	26.67	26.67
Average	Our method	**60.30**	**57.32**	**58.68**
	$\mathbf{f_1}$ as in [24]	23.61	24.35	23.93

B. Test B: Homogeneous dataset with noise

In this test, we consider a dataset consisting of a task execution in standard conditions learned from humans with DMPs, and nine more executions generated adding low-frequency noise to the kinematic signature. This generates an homogeneous dataset with noise, which increases the kinematic variability between different users and requires additional robustness. Noise is generated with power frequency spectrum:

$$S(f) = \beta \frac{1}{f^{\lambda}}$$

with $\lambda = 7.5$ so that frequencies above the fundamental frequency $1.5\,\mathrm{Hz}$ of human hand [28] have power below 0.05β, hence they can be neglected. β is related to the noise variance. We vary it in the range $[0.01, 0.09]$ with a step of 0.01 to generate the synthetic executions. In Table IV we evaluate the matching score considering the full synthetic dataset. The average matching score over all actions with our Algorithm 1 improves the results of [24]. We then evaluate the performance of our action classification algorithm in Table V. We choose $k = 36$ (number of occurrencies of `release` action) for k-NN classification. Results are comparable with Test A, though the improvement with respect to [24] is significant only for short actions.

TABLE IV: Matching scores for Test B. All values are percentages.

Action		Matching score
`move(A,ring,C)`		94.78
`move(A,peg,C)`		90.47
`move(A,center,C)`		97.53
`grasp(A,ring,C)`		76.02
`extract(A,ring,C)`		71.80
`release(A)`		93.18
Average	Our method	**87.30**
	Method in [24]	81.90

TABLE V: Action classification results for Test B. All values are percentages.

Action	Feature array	Pr	Rec	$F1$
`move(A,ring,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	75.00	75.00	75.00
	$\mathbf{f_1}$ as in [24]	75.00	75.00	75.00
`move(A,peg,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	100.00	100.00	100.00
	$\mathbf{f_1}$ as in [24]	100.00	100.00	100.00
`move(A,center,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	40.82	**55.56**	47.06
	$\mathbf{f_1}$ as in [24]	**45.00**	50.00	**47.37**
`grasp(A,ring,C)`	$[\mathbf{f_2}, \mathbf{f_3}]$	**40.00**	**50.00**	**44.44**
	$\mathbf{f_1}$ as in [24]	17.78	22.22	19.66
`extract(A,ring,C)`	$[\mathbf{f_2}, \mathbf{f_3}]$	**51.02**	**69.44**	**58.82**
	$\mathbf{f_1}$ as in [24]	47.37	50.00	48.65
`release(A)`	$[\mathbf{f_2}, \mathbf{f_3}]$	**40.00**	**40.00**	**40.00**
	$\mathbf{f_1}$ as in [24]	26.67	26.67	26.67
Average	Our method	**57.81**	**65.00**	**60.89**
	$\mathbf{f_1}$ as in [24]	51.97	53.98	52.89

C. Test C: Non-homogeneous dataset

In Test C we consider a dataset of only 4 executions under non-homogeneous environmental conditions, i.e. the initial conditions of the setup and the action sequence vary between executions. One execution is in standard environmental conditions (34 actions), while 3 other ones are depicted in Figure 3 and include a failure condition with 18 actions (the

TABLE VI: Matching scores for Test C. All values are percentages.

Action		**Matching score**
`move(A,ring,C)`		95.36
`move(A,peg,C)`		92.83
`move(A,center,C)`		96.35
`grasp(A,ring,C)`		78.40
`extract(A,ring,C)`		83.08
`release(A)`		85.45
Average	Our method	**88.58**
	Method in [24]	81.90

TABLE VII: Action classification results for Test C. All values are percentages.

Action	**Feature array**	Pr	Rec	$F1$
`move(A,ring,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	81.82	90.00	85.72
	$\mathbf{f_1}$ as in [24]	**90.00**	90.00	**90.00**
`move(A,peg,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	80.00	80.00	80.00
	$\mathbf{f_1}$ as in [24]	80.00	80.00	80.00
`move(A,center,C)`	$[\mathbf{f_1}, \mathbf{f_2}, \mathbf{f_3}]$	**40.00**	40.00	**40.00**
	$\mathbf{f_1}$ as in [24]	22.22	40.00	28.57
`grasp(A,ring,C)`	$[\mathbf{f_2}, \mathbf{f_3}]$	**60.00**	75.00	**66.66**
	$\mathbf{f_1}$ as in [24]	56.25	75.00	64.29
`extract(A,ring,C)`	$[\mathbf{f_2}, \mathbf{f_3}]$	**100.00**	**62.50**	**76.92**
	$\mathbf{f_1}$ as in [24]	12.50	12.50	12.50
`release(A)`	$[\mathbf{f_2}, \mathbf{f_3}]$	52.38	52.38	52.38
	$\mathbf{f_1}$ as in [24]	52.38	52.38	52.38
Average	Our method	**69.03**	**66.65**	**66.95**
	$\mathbf{f_1}$ as in [24]	52.23	58.31	54.62

ring falls during the execution, Figure 3a), an unconventional scenario with occupied colored pegs involving 17 actions 3b, and an execution with PSMs moving simultaneously 3c including 12 actions. Actions in each execution repeat multiple times. This is a challenging dataset for unsupervised action identification, since not all actions appear in all executions, and both kinematic and semantic features do not repeat in the same way over the whole dataset. Table VI shows that the matching score is comparable with Test B both for each action and on average, and higher than [24] on average. Results of k-NN classification with $k = 21$ (number of occurrencies of `release` action) are shown in Table VII. Our method outperforms the scores in [24] (average F1-score rises from 54% to almost 67%). The results for `extract` action are significantly improved, reaching precision of 100% and F1-score of 77% (both scores reach only 12% on the dataset of [24]). Also F1-score and precision (almost double) for `move(A,center,C)` are better. This is even more significant considering that this action is the least frequent in the dataset, appearing only in two scenarios (once in Figure 3a and 4 times under standard environmental conditions). A slight decrease in the performance is only recorded for `move(A,ring,C)`, though the F1-score is the highest among all actions.

D. Computational performance

The computational performance of our action identification algorithm is tested on a PC using 2.6 GHz Intel Core i7-6700HQ CPU (4 cores / 8 threads). The full action identification procedure (segmentation + classification) for one execution trace requires $0.45\,\mathrm{s}$ on average (maximum $0.58\,\mathrm{s}$ for the standard execution with the highest number of actions). For changepoint filtering, the time required for fluent identification from video frames must be also accounted for. Fluent identification from a single frame requires $0.09\,\mathrm{s}$ on average; thus, fluent identification in a full execution trace requires at most $2.88\,\mathrm{s}$ for the execution with most actions (standard environmental conditions). This results in better performance than [24], where $5\,\mathrm{s}$ are needed for action identification using a better CPU (3.4 GHz Intel Core i7-3770 with 4 cores / 8 threads). In fact, one main limitation of [24] lies in the use of persistence analysis [33] and dynamic time warping (DTW) [39] to identify changepoints. Persistence analysis removes noise and identifies relevant local minima and maxima in the kinematic features, but still results in spurious changepoints which must be removed with further post-processing through DTW based on Euclidean distance. DTW evaluates the alignment of *all possible* consecutive segments in the kinematic signature, including spurious ones. As also claimed by the autrhors of [24], this increases computational complexity. Notice that the computational performance also depends on the number of segments to be identified in execution traces. In [24], 12 actions are involved in a single execution trace (with possibly some repetitions), while the highest number of segments in our execution trace under standard environmental conditions is 32; hence, the comparison of computational performance is fair.

The computational performance of our algorithm is also comparable with TSC [19] ($\approx 1 - 10$ s, though the authors do not provide hardware information and do not consider automatic extraction of visual features, presented as a future work).

IV. Conclusion

Unsupervised segmentation of surgical actions from expert executions is crucial for robotic surgery. In this paper, we presented a novel algorithm combining kinematic and semantic visual features to identify actions in unlabeled executions of a training ring transfer task with dVRK. We addressed some limitations of state-of-the-art research, as identification from datasets including short actions, few executions of the task and possibly non-homogeneous (anomalous) flows of actions. We validated our algorithm in three different experimental conditions. We first evaluated the performance on a limited dataset of 9 executions from humans with different expertise in using dVRK. We were able to significantly improve results in the state of the art under different experimental conditions (but considering only expert executions), doubling F1-score also for short actions.

We then tested the robustness of our algorithm on a similar dataset, where a single human execution was augmented with low-frequency kinematic noise (10 executions total). Results were comparable with the previous test, though the improvement with respect to the state of the art was not similarly relevant.

Finally, we showed that our algorithm is able to improve the state of the art (especially for short actions) on a non-homogeneous dataset including only 4 executions and actions appearing rarely (only 2 times).

The average F1-score over all actions is $58 - 66\%$, comparable or better than state-of-the-art methods based on deep learning [23] (51%) for a similar task to ours. Improvements mainly rise from the inclusion of semantic visual features for classification, which compensate kinematic variability.

We also showed that our algorithm has comparable or better computational performance than state of the art, allowing feature extraction and action identification within few seconds.

This paper represents the first step towards unsupervised learning of action-level SPMs. Next research will combine our algorithm with inductive learning of interpretable SPMs [40], [41], and unsupervised semantic recognition of events from raw videos [42], [43].

REFERENCES

[1] J. H. Palep, "Robotic assisted minimally invasive surgery," *Journal of minimal access surgery*, vol. 5, no. 1, p. 1, 2009.

[2] B. Weksler *et al.*, "Robot-assisted minimally invasive esophagectomy is equivalent to thoracoscopic minimally invasive esophagectomy," *Diseases of the Esophagus*, vol. 25, no. 5, pp. 403–409, 2012.

[3] M. Daouadi *et al.*, "Robot-assisted minimally invasive distal pancreatectomy is superior to the laparoscopic technique," *Annals of surgery*, vol. 257, no. 1, pp. 128–132, 2013.

[4] R. Bharathan *et al.*, "Operating room of the future," *Best Practice & Research Clinical Obstetrics & Gynaecology*, vol. 27, no. 3, pp. 311–322, 2013.

[5] T. D. Nagy and T. Haidegger, "A dvrk-based framework for surgical subtask automation," *Acta Polytechnica Hungarica*, pp. 61–78, 2019.

[6] G. De Rossi *et al.*, "A First Evaluation of a Multi-Modal Learning System to Control Surgical Assistant Robots via Action Segmentation," *IEEE Transactions on Medical Robotics and Bionics*, pp. 1–11, 2021.

[7] R. Muradore *et al.*, "Development of a cognitive robotic system for simple surgical tasks," *Int J of Advanced Robotic Systems*, vol. 12, no. 4, p. 37, 2015.

[8] M. Ginesi *et al.*, "A knowledge-based framework for task automation in surgery," in *2019 19th International Conference on Advanced Robotics (ICAR)*, Dec 2019, pp. 37–42.

[9] M. Ginesi *et al.*, "Autonomous task planning and situation awareness in robotic surgery," in *2020 IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3144–3150.

[10] F. Lalys and P. Jannin, "Surgical process modeling: a review," *International J CARS*, vol. 9, no. 3, pp. 495–511, 2014.

[11] K. Charrière *et al.*, "Real-time analysis of cataract surgery videos using statistical models," *Multimedia Tools and Applications*, vol. 76, no. 21, pp. 22 473–22 491, 2017.

[12] A. P. Twinanda *et al.*, "Endonet: a deep architecture for recognition tasks on laparoscopic videos," *IEEE transactions on medical imaging*, vol. 36, no. 1, pp. 86–97, 2016.

[13] O. Dergachyova *et al.*, "Knowledge transfer for surgical activity prediction," *International journal of computer assisted radiology and surgery*, vol. 13, no. 9, pp. 1409–1417, 2018.

[14] S.-R. Ke *et al.*, "A review on video-based human activity recognition," *Computers*, vol. 2, no. 2, pp. 88–131, 2013.

[15] A. Naftel and S. Khalid, "Classification and prediction of motion trajectories using spatiotemporal approximations," in *Proc. Int. Workshop on Human Activity Recognition and Modelling*, 2005, pp. 17–26.

[16] A. Murali *et al.*, "Tsc-dl: Unsupervised trajectory segmentation of multi-modal surgical demonstrations with deep learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4150–4157.

[17] Z. Shao *et al.*, "Unsupervised trajectory segmentation and promoting of multi-modal surgical demonstrations," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 777–782.

[18] Y. Gao *et al.*, "Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling," in *MICCAI Workshop: M2CAI*, vol. 3, 2014, p. 3.

[19] S. Krishnan *et al.*, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," in *Robotics Research*. Springer, 2018, pp. 91–110.

[20] M. J. Fard *et al.*, "Soft boundary approach for unsupervised gesture segmentation in robotic-assisted surgery," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 171–178, 2016.

[21] B. van Amsterdam *et al.*, "Weakly supervised recognition of surgical gestures," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9565–9571.

[22] N. J. Soper and G. M. Fried, "The fundamentals of laparoscopic surgery: its time has come." *Bull Am Coll Surg*, vol. 93, no. 9, pp. 30–32, 2008.

[23] X. Shi *et al.*, "Domain adaptive robotic gesture recognition with unsupervised kinematic-visual data alignment," *arXiv preprint arXiv:2103.04075*, 2021.

[24] F. Despinoy *et al.*, "Unsupervised trajectory segmentation for surgical gesture recognition in robotic training," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 6, pp. 1280–1291, 2015.

[25] U. Akdemir *et al.*, "An ontology based approach for activity recognition from video," in *Proceedings of the 16th ACM international conference on Multimedia*, 2008, pp. 709–712.

[26] A. Roberti *et al.*, "Improving rigid 3-d calibration for robotic surgery," *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 4, pp. 569–573, 2020.

[27] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[28] I. D. Loram *et al.*, "The frequency of human, manual adjustments in balancing an inverted pendulum is constrained by intrinsic physiological factors," *The Journal of physiology*, vol. 577, no. 1, pp. 417–432, 2006.

[29] R. W. Schafer, "What is a savitzky-golay filter? [lecture notes]," *IEEE Signal Processing Magazine*, vol. 28, no. 4, pp. 111–117, 2011.

[30] X. Wang, "Three-dimensional kinematic analysis of influence of hand orientation and joint limits on the control of arm postures and movements," *Biological Cybernetics*, vol. 80, no. 6, pp. 449–463, 1999.

[31] T. Pistohl *et al.*, "Prediction of arm movement trajectories from ecog-recordings in humans," *Journal of neuroscience methods*, vol. 167, no. 1, pp. 105–114, 2008.

[32] C. Truong *et al.*, "Selective review of offline change point detection methods," *Signal Processing*, vol. 167, p. 107299, 2020.

[33] H. Edelsbrunner *et al.*, "Topological persistence and simplification," in *Proceedings 41st annual symposium on foundations of computer science*. IEEE, 2000, pp. 454–463.

[34] J. L. Crowley *et al.*, "Context aware vision using image-based active recognition," 2004.

[35] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[36] M. Ginesi *et al.*, "Dynamic movement primitives: Volumetric obstacle avoidance," in *2019 19th International Conference on Advanced Robotics (ICAR)*, Dec 2019, pp. 234–239.

[37] M. Ginesi *et al.*, "Dynamic movement primitives: volumetric obstacle avoidance using dynamic potential functions," *Journal of Intelligent and Robotic Systems*, vol. 101, no. 4, p. 20, 2021.

[38] M. Ginesi, N. Sansonetto, and P. Fiorini, "Overcoming some drawbacks of dynamic movement primitives," *Robotics and Autonomous Systems*, p. 103844, 2021.

[39] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[40] D. Meli *et al.*, "Towards inductive learning of surgical task knowledge: a preliminary case study of the peg transfer task," *Procedia Computer Science*, vol. 176, pp. 440–449, 2020.

[41] D. Meli, M. Sridharan, and P. Fiorini, "Inductive learning of answer set programs for autonomous surgical task planning," *Machine Learning*, pp. 1–25, 2021.

[42] C. Gan *et al.*, "Concepts not alone: Exploring pairwise relationships for zero-shot video activity recognition," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[43] M. Sridhar *et al.*, "Unsupervised learning of event classes from video," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 2010, pp. 1631–1638.