Unsupervised Trajectory Segmentation and Gesture Recognition through Curvature Analysis and the Levenshtein Distance

Guillem Tapia, Adrià Colomé, Carme Torras
Institut de Robòtica i Informàtica Industrial (CSIC-UPC). Spain
guillem.tapia@estudiantat.upc.edu, {acolome,torras}@iri.upc.edu

***Abstract*—This work discusses segmentation and gesture recognition in human-driven robotic trajectories, a technique with applications in several sectors such as in robot-assisted minimally invasive surgery (RMIS) training. By decomposing entire movements -gestures- into smaller actions -sub-gestures-, we can address gesture recognition accurately. This paper extends a bottom-up approach used in surgical gesture segmentation and incorporates natural language processing (NLP) techniques to match sub-gestures with letters and treating gestures as words. We evaluated our algorithm using two different datasets with trajectories on 2D and 3D. This NLP-inspired model obtains an average F1-score of 94.25% in the segmentation tasks, an accuracy of 87.05% in the learning stage, and an overall accuracy of 88.79% in the fully automated execution. These results indicate that the method effectively identifies and interprets new surgical gestures autonomously without the need for human intervention.**

I. Introduction

Segmentation and gesture recognition in human-driven robot motion allow to identify and classify distinct movements. In applications such as robot-assisted minimally invasive surgery (RMIS), it has been widely used in areas like automated surgical competence and skill assessment [1] [2] and adaptive surgical training [3] to quantify the surgical process without the need for expert monitoring. This paper develops and evaluates a method for an efficient and fast segmentation and recognition of meaningful gestures generated by human driven robots.

To do so, we extend the bottom-up approach to surgical gesture segmentation and recognition described in [4]. A gesture is defined as an atomic unit of intentional task activity resulting in a perceivable and meaningful outcome (e.g., grabbing an object). Every gesture is made up of a set of primitives called sub-gestures, which are numerical representations of the component sub-gestures needed to carry out a gesture. Sub-gestures, such as ”go towards” or ”turn left,” are single-handed motions without any semantic background.

In this paper, we propose a novel approach that includes natural language processing (NLP) methods. We pair sub-gesture clusters with lowercase letters considering the gesture as a word, where each word -gesture- is then composed of a series of concatenated characters -sub-gestures-. Therefore, we use word similarity measures from NLP to identify similar gestures by the words their segments form. This NLP-inspired model facilitates interpretation and analysis by using methods such as those employed in word processing for better identification and classification of gestures.

The remaining sections are structured as follows. In Section II, we first decompose robotic trajectories into sub-gestures through an unsupervised segmentation method based on the search for abrupt changes. Given the segmented sub-gestures, we parameterize and cluster them by labelling them as characters and describing gestures as words by the concatenation of the characters in Sections III-A and III-B. In Section III-C, we use word similarity measures from NLP to classify gestures into reference gestures previously defined on a dictionary. Later, the datasets and evaluation methods used are detailed in Section IV. Finally, the results are shown in Section V and the conclusions drawn from this work are presented in Section VI.

II. Unsupervised Sub-gesture Segmentation

We propose a segmentation approach consisting of four distinct steps. The first step uses the input kinematic data to generate additional 3D-invariant kinematic signals and select the relevant metrics to define the principal and dissimilarity signals matrices. It finds relevant timestamps for temporal sub-gesture decomposition in these signals in the next step. After that, in the third step, temporal and spatial dissimilarities are used to assess the importance of timestamps. In order to produce the most relevant sub-gesture segmentation, the fourth step ultimately chooses the most appropriate timestamps according to the dissimilarity ranking and a minimum segment filter.

A. Data Preprocessing

The input data consists of a temporal series of end-effector trajectories, with positions in the 2-dimensional or 3-dimensional Cartesian space.

In order to reduce measurement noise and record only voluntary motions, a band-stop filter with low and high frequencies set as 3Hz and 9Hz respectively is applied [5].

1) Definition of Principal Signals Matrix and Dissimilarity Signals Matrix: We propose a new approach where we define two signatures S_P, S_D. Then, selecting the best metrics for each process simplifies and improves the method.

First, we define the principal signature S_P, which represents the trajectory at each point with a particular metric and will be used to propose the initial points of separation of the subgestures by their analysis. For its definition, the optimal signals are selected. In the most general case, it could be defined by the metrics formed by each point along the trajectory of the following kinematic variables of the tooltip: position $\mathbf{r}(t) = \{X(t),$ $Y(t),$ $Z(t)\}$, orientation $\mathbf{q}(t) = \{Q_w(t),$ $Q_x(t),$ $Q_y(t),$ $Q_z(t)\}$, position curvature $\kappa(t)$, orientation curvature $\Omega(t)$ and their first order derivatives wrt. time $\dot{\kappa}(t)$ and $\dot{\Omega}(t)$.

$$S_P = \{\mathbf{r}(t), \mathbf{q}(t), \kappa(t), \Omega(t), \dot{\kappa}(t), \dot{\Omega}(t)\}$$

However, in our study we only consider the position curvature first derivative wrt. time $\dot{\kappa}(t)$ to separate the sub-gestures of a trajectory, which is computationally more efficient and we have empirically found that it is the most informative term. We use the absolute values of S_P to allow for a better analysis while avoiding duplicates in cutting trajectories into sub-gestures. The latter because when there is a peak on curvature, the curvature derivative has two peaks (the positive and the negative) but it is desirable to represent only one abrupt change. Then, the principal signature S_P is defined as follows:

$$S_P = |\dot{\kappa}(t)|, \tag{1}$$

where $\kappa(t)$ defines the curvature on each point of the trajectory. $\kappa(t)$ and its first derivative wrt. time $\dot{\kappa}(t)(t)$ can be obtained [6] by:

$$\kappa(t) = \frac{\|\dot{\mathbf{r}}(t) \times \ddot{\mathbf{r}}(t)\|}{\|\dot{\mathbf{r}}(t)\|^3} \tag{2}$$

$$\dot{\kappa}(t) = \frac{d\kappa(t)}{dt} \tag{3}$$

where $\mathbf{r}(t) = (x(t), y(t), z(t))$ is a 3D trajectory parameterized by t, and $\dot{\mathbf{r}}(t)$ and $\ddot{\mathbf{r}}(t)$ are the first and second derivatives of $\mathbf{r}(t)$ with respect to t.

Later, the best segmentation candidates will be selected according to the dissimilarity ranking that will be defined in II-C. These dissimilarities will be computed using a new signature called dissimilarity signature S_D, which also represents the trajectory at each point with new particular metrics. While [4] uses the same signature vector including position/orientation and curvature, torsion and their derivatives, this study focuses on considering only the Cartesian positions of the tooltip for the dissimilarity signature S_D for capturing the notion of dissimilarity in a more data-efficient manner, i.e.:

$$S_D = \{X(t), Y(t), Z(t)\}$$

Therefore, both signatures of interest S_P and S_D are computed and stored as matrices.

B. Initial Candidates Selection

Each sub-gesture is defined by a first and final point that delimits it. The goal now is to detect when the operator performs a change of sub-gesture. We tackle this problem by identifying this change finding abrupt turns on the trajectory. These turns can be captured by studying the curvature of the trajectories. However, the interest does not fall in whether the curvatures present either high or low values, but rather on those points of the trajectory where there is a strong change of the curvature.

Then, the timestamps in which the operator makes an abrupt change of direction either by changing to a higher or lower curvature are identified. Other studies use the notion of persistence to detect these change points [7] [8] [9]. This method can be related in some way to the notion of persistence homology, since also segments the trajectory according to the change of the signal on time.

Here, we follow a similar methodology as that in [10]. The original algorithm segments the trajectory around which it curls detecting the points where the curvature is large compared to most of the trajectory. In our approach, we use the same segmentation algorithm but now using the first derivative of the curvature instead of the curvature.

Also, two additional considerations are imposed: a clipping value and a minimum filtration value. It determines the maximum magnitude allowed for signals included on S_P from (1). Setting an appropriate clipping value prevents extreme values from distorting the selection of the threshold on the presented algorithm. Here, an appropriate clipping value for the derivative of the curvature was empirically set to 300, as larger values are considered outliers. At the same time, a minimum filtration value of 30 is applied to avoid over-selection when curvatures are low.

C. Dissimilarity Ranking

Now, we evaluate the dissimilarity of the extracted candidate points by using the dissimilarity signature S_D

to determine their significance as potential solutions for delineating a sub-gesture. Other studies compare the use of Hausdorff, Fréchet and Dynamic Time Warping distances for similar purposes [4]. In our study, we use Dynamic Time Warping (DTW) as it is proven to be the most effective. It is used to quantify the shape and temporal fluctuations of consecutive segments. Our analysis quantifies this dissimilarity between the segments divided by the proposed cutting points using the dissimilarity signature S_D.

Dynamic Time Warping (DTW) is a technique for aligning and comparing temporal sequences while accounting for differences in speed and timing. It can be formulated as an optimization problem to find the best alignment between two sequences A and B:

$$DTW(P, Q) = \min_{\text{path}} \sum_{(i,j)\in\text{path}} d(p_i, q_j), \tag{4}$$

where $d(p_i, q_j)$ represents the local distance between elements p_i and q_j of sequences P and Q, respectively. The minimum is taken over possible alignments (paths).

D. Minimum Segment Filter

Now, the best candidate segmentation timestamps are selected according to the dissimilarity ranking. In our study, we only apply a filter that discards segments with a length below a predefined threshold. In such cases, we keep as a candidate the edge of such segments that has the highest dissimilarity distance. This process discards those candidates trying to detect the same change of sub-gesture. Although a pairwise Non-Maximum Suppression (NMS) procedure could be also used to filter the best candidates like in [4], we found that to be inefficient as it eliminated too many candidate points.

III. Learning and Recognition

In this section, we introduce our NLP modelling approach. Every segmented sub-gesture is converted into a vector of parameters, that are then clustered into groups representing the types of sub-gestures. Finally, we recognize our gesture classifying them into reference gestures previously defined on a dictionary using word similarity measures from NLP.

A. NLP modelling

In order to recognise the gestures, we gather the segmented parts (sub-gestures) within the trajectories in the dataset and generate clusters of sub-gestures. These clusters will form an alphabet in which each *letter* is a gesture part, and a concatenation of these letters can form words - gestures -. We can then identify gestures by matching their *letter* sequences to those on a dictionary, with methodologies used in NLP. In particular, we used the Levenshtein distance.

B. Sub-gesture Learning

1) Feature transformation: After segmenting a trajectory - gesture - into a sequence of sub-gestures, every sub-gesture is encoded as a parameter vector. The first metric to study is the orientation of the trajectory wrt. our reference basis, the canonical Cartesian coordinate system. Also, by using principal component analysis (PCA) with the trajectory points, we can determine the local frame of the trajectory and align it with the canonical frame. The eigenvectors of such PCA decomposition V=[v_1,v_2,v_3] represent the principal direction of the covariance ellipsoid of the trajectory points [11]. In this variation of the PCA, our first vector w_1 of the basis is imposed to be the vector that goes from the initial to the final point of the trajectory instead of its first principal component vector. This change is done in order to encapsulate the direction of the trajectory. Then, we build a base W by setting w_2 as the vector orthonormal to w_1 and within the plane formed by v_1, v_2. Last, $w_3 = w_1 \times w_2$. From here, the rotation matrix R that transforms W to our reference basis is computed. Defining the reference basis O as the identity matrix,

$$R = OW^{-1} = W^{-1} \tag{5}$$

Then, the ZYX Euler angles ϕ, θ, ψ are obtained from the rotation matrix R, which will be the first parameters of our characterizing vector and express the orientation of the trajectory wrt. the canonical base. For the analysis that follows, we rotate trajectory to our reference basis, decontextualizing orientation from now on and creating a standardized framework for analyzing and interpreting trajectory new parameters across various datasets and contexts.

Next, L_x is the greatest value along the x-axis and represents the horizontal length on our new first axis of the trajectory (the length of the trajectory along direction w_1). Once captured, the L_x is again decontextualized from trajectory by normalizing it to a length of 1 to compare trajectories of different sizes.

L_y then represents the largest value along the y-axis and provides information about the height of the trajectory, along direction w_2. From here on, trajectories will not be decontextualized, as the following parameters are no longer affected by the previous ones.

The x-value (in w_1 direction) at which the largest peak L_y is located is then represented as M_{xy}. It quantifies the location of the curve peak and indicates the asymmetry of the motion wrt. its center.

Then, the area created by the trajectory in the xy plane is denoted by A_{xy}. This parameter uses the area covered by the projection of the trajectory on the xy plane to determine the absolute width of the curve. this is intended to capture the absolute but not relative width.

Finally, the area created by the trajectory in the xz plane denoted by A_{xz} is also included to quantify the lateral motion. This metric will be the only one related to the z-axis because we want this axis to have an existing but small impact on the characterization of the sub-gestures. As we have already performed a kind of PCA, our z-axis represents the w_3 direction with the smallest variance, and therefore, the least significant when distinguishing our samples.

Therefore, the complete sub-gesture approximation vector is $\mathbf{d} = \{\phi, \theta, \psi, L_x, L_y, M_{xy}, A_{xy}, A_{xz}\}$, while for a 2D case, d is reduced from eight to five dimensions $\mathbf{d} = \{\psi, L_x, L_y, M_{xy}, A_{xy}\}$.

2) GMMs clustering: After converting each subgesture l of each trajectory m, g_l^m, into a vector of parameters $\mathbf{d}_l^m$, we now propose to collect all these vectors of parameters and cluster them into what will become the *letters* of our alphabet. To do so, we applied Gaussian Mixture Models (GMMs) [12]. GMM fits a multi-modal Gaussian distribution on the parameters $\mathbf{d}$ by fitting the mixing weights π_k, and the mean and covariance of each mode $k = 1..K$ of the distribution, $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$, which will be the K *letters* in our alphabet of sub-gestures:

$$p(\mathbf{d}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{d}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

We introduce the data of all our parameterized sub-gestures and define the number of clusters empirically depending on the complexity of the dataset. As a result, similar sub-gestures are grouped into clusters. Still, miss-classification errors may occur. In this study, two similarity approaches are studied to assess the miss-classification: the cluster-to-cluster and the point-to-cluster similarity approaches.

3) Cluster-to-cluster similarity: We created a similarity matrix to define how similar these clusters are and thus how likely it is that a sub-gesture has been misclassified into one cluster rather than another.

Firstly, the Kullback-Leibler (KL) divergence, D_{KL} [13] is used to measure how far are the clusters between them. The KL divergence is a dissimilarity measure that quantifies how good a probability distribution approximates another one. In this work, it is used to assess the how far are two clusters between them.

The next step is the transformation of the divergence to an inverse magnitude such as the similarity to try to capture how probable it is for a point priory classified on one cluster to belong to another specific cluster truly. Firstly, we construct a matrix D with each entry D_{k_i,k_j} representing the KL non-symmetric divergence between every pair of clusters (k_i) and (k_j).

$$D_{k_i,k_j} = \mathrm{D}_{KL}(k_i \| k_j) \tag{6}$$

To ensure that the divergences are within an acceptable range, we normalize the matrix D and use a similarity matrix $\hat{S}$, obtained by the negative exponential of D as $\hat{S}_{k_i,k_j} = e^{-D_{k_i,k_j}/\lambda}$, where λ is a normalization factor. We then regularize $\hat{S}$ by the sums of the values of each similarity row. This is because the sum of the odds that a sample is part of a cluster should sum to one, in a similar way as if they were real probabilities:

$$S_{k_i,k_j} = \frac{\hat{S}_{k_i,k_j}}{\sum_{j=1}^{N} \hat{S}_{k_i,k_j}}.$$

With this expression, closer clusters obtain higher similarity scores. We set the γ parameter so that the average probability that a point found in a cluster actually belongs to it is equal to a preset threshold.

$$\gamma = \frac{1}{N} \sum_{i=1}^{N} S_{k_i,k_i}$$

The preset thresholds can be modified depending on whether the classification is expected to be very reliable or not. The higher the expected reliability, the higher the lambda term should be. The preset threshold by default is defined empirically as $\gamma = 0.7$.

4) Point-to-cluster similarity: As an alternative cluster-to-cluster similarity, we study the use of point-to-point similarity by evaluating the probability of individual data entries belonging to every cluster. The cluster-to-point correspondence provided by the clustering process allows to define a probabilistic association between every point and every cluster. While the previous cluster-to-cluster measure needs to be normalized to interpret the divergences as probability, the point-to-cluster probabilities are a direct outcome of the GMM clustering and are inherent as true conditional probabilities and thus do not need to be rescaled. The probability of a point $\mathbf{d}$ belonging to cluster k in the GMM framework is:

$$p(k|\mathbf{d}) = \frac{\pi_k \mathcal{N}(\mathbf{d}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)},$$

where $p(k|\mathbf{d})$ is the posterior probability that point $\mathbf{d}$ belongs to cluster k. Analogous to the previous cluster-to-cluster, the similarity matrix is created for collecting all cross probabilities, denoted $S_{x,k} = p(k|\mathbf{d})$.

On the comparison of the two presented methods, a point-to-cluster approach gives probabilities as an output rather than ambiguous divergences, hence enabling users to interpret outcomes better, reducing the need for further normalizing. However, in the individual evaluation the accuracy strongly depends on how well the real data fits the GMMs model. Nevertheless, the cluster-to-cluster approach has a reduced computational load.

C. Gesture Recognition

As in other works, [14] [15] [16], we use a dictionary-based approach that uses a pre-defined set of gestures where the letter sequences of these reference dictionary words are learned through a training set.

1) Gesture Vocabulary and Dictionary Definition: A key feature of the process is the definition of a dictionary, a collection of the possible tasks to do to classify gestures into several types. In our study, four gestures/tasks, which be explained later on IV-A, have been contemplated: S-shaped, U-shaped, Place-and-place and Eating soup trajectories.

The next step is to learn which concatenation of sub-gestures represents each of the presented gestures. Firstly, we annotate what gestures are each trajectory from the training sets. Then, the reference gesture dictionary is constructed by concatenating the predominant sub-gesture clusters from a training set. In other words, the most frequent concatenation of letters is assigned as the dictionary word for each gesture. Hence, we obtain a reference dictionary of gestures where each type of gesture is defined by a word.

2) Gesture Classification: For the recognition step, the objective is to recognize new gestures/trajectories by classifying them in the correct gesture of the dictionary. Firstly, an estimation of the concatenation of sub-gestures is done by the segmentation, the sub-gesture learning and classification. Secondly, these estimated gestures already coded as a word are compared to all the words of the dictionary. The gesture is classified as the most similar word, keeping the predicted gesture and the similarity measurement associated as output.

For the comparison of gestures or words, the Levenshtein distance is used. It measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into another. The Levenshtein distance between strings G and H of lengths $|G|$ and $|H|$ respectively is the following.

$$\mathrm{lev}_{G,H}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j)=0, \\ \min \begin{cases} \mathrm{lev}_{G,H}(i-1,j)+1, \\ \mathrm{lev}_{G,H}(i,j-1)+1, \\ \mathrm{lev}_{G,H}(i-1,j-1)+1_{(G_i \neq H_j)} \end{cases} & \text{otherwise.} \end{cases} , \tag{7}$$

where $1_{(G_i \neq H_j)}$ is the indicator function which equals 0 when $G_i = H_j$, and 1 otherwise.

In our study, we introduce an algorithm that allows the user to specify different weights for substitution. On the cluster-to-cluster similarity approach, it depends on the involved clusters. For instance, substituting 'a' for 'b' should be less expensive than substituting 'a' for 'c'. The previous matrix with cross similarities S derived from Kullback-Leibler divergence determines these costs. The substitution cost from letter a to letter b becomes $(1 - S_{a,b})$. On the point-to-cluster approach, the cost of substitution of that particular letter a being the sub-gesture $\mathbf{d}$ to letter b is equal to $(1 - S_{\mathbf{d},b})$. For this purpose, let us denote GL as variable that collects the labels of the sub-gestures involved on the string G. Algorithm 1 summarizes the process described.

Algorithm 1 Levenshtein Distance Calculation

```
1: Input: $G, GL, H, S$
2: Output: $dp[i][j]$
3: $m = \text{len}(G)$
4: $n = \text{len}(H)$
   Initialize the dynamic programming matrix dp of size $(m+1) \times (n+1)$ with all elements as $0$
5: for $i = 0$ to $m$ do
6:     $dp[i][0] = i$
7: end for
8: for $j = 0$ to $n$ do
9:     $dp[0][j] = j$
10: end for
11: for $i = 1$ to $m$ do
12:     for $j = 1$ to $n$ do
        Compute costs of insertion, deletion and substitution considering general weights and similarities for substitutions
13:         $cost_{ins} = dp[i][j-1]$
14:         $cost_{del} = dp[i-1][j]$
15:         if Cluster-to-cluster similarity then
16:             $cost_{subs} = dp[i-1][j-1] + (1 - S_{G[i-1],H[j-1]})$
17:         else if Point-to-cluster similarity then
18:             $cost_{subs} = dp[i-1][j-1] + (1 - S_{GL[i-1],H[j-1]})$
19:         end if
        Select the minimum cost among insertion, deletion, and substitution and store it in the matrix
20:         $dp[i][j] = min(cost_{ins}, cost_{del}, cost_{subs})$
21:     end for
22: end for
   Return the edit distance between the strings
23: return $dp[m][n]$
```

IV. Datasets and Evaluation

A. Datasets

1) Letters Dataset: This dataset consists on trajectories characters from A to Z in a 2D space handwritten by an end-effector. In each trajectory only one letter is drawn. There are ten attempts for each letter made by the same person. Kinematic data - Cartesian positions on 2D space - are captured at a frequency of 100 Hz.

2) 3D Trajectories Dataset: This dataset consists of a collection of trajectories performing different tasks recording end-effectors without opening or closing the grippers. Their kinematic data - Cartesian positions on 3D space - are captured at frequency of 20 Hz.

S-shaped trajectory. This dataset contains trajectories that follow an S-shaped pattern. In this dataset, there are four attempts executed by the same person.

U-shaped trajectory. This dataset comprises trajectories that form a U-shaped pattern. In this sample, there are seven attempts executed by the same person.

Place-and-place trajectory. This dataset includes trajectories that simulate a task of placing an object from one location and to another, and then into another. Here, there are seven attempts executed by the same person.

Eating Soup trajectory. Formed by trajectories recorded while feeding a person soup with a spoon. There are twenty attempts executed by the same person.

B. Evaluation

We evaluate our method by comparing the desired results (ground truth annotations) with those obtained through standard classification metrics such as accuracy, precision, recall and F1-score. We denote True/False (T/N) Positives/Negatives (P/N), and accuracy is then defined as the number of instances that have been predicted correctly (N_c) compared to all frames (N).

$$\text{Accuracy} = \frac{N_c}{N} \tag{8}$$

In addition, precision ($\frac{TP}{TP+FP}$), recall ($\frac{TP}{TP+FN}$, observations from a single class correctly identified), and F1-score ($F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$) are also used.

We evaluated segmentation, sub-gesture learning, and gesture classification separately. For segmentation comparison, timestamps of the desired segmentation are first annotated. To evaluate the sub-gesture learning stage, the expected clusters with which the sub-gestures should be classified are compared using a leave-one-out cross-validation process (LOOCV). For this purpose, each observation is considered as the validation set and the rest of the observations as the training sets. From these training sets, the reference gesture dictionary is constructed by concatenating the predominant sub-gesture clusters. If the sub-gesture cluster to be validated differs from the one collected in the reference dictionary, it is counted as an error. To evaluate gesture classification, the expected gesture is also noted. Then, the final classification of the gesture is compared again for correctness through a leave-one-out cross-validation process.

V. Results

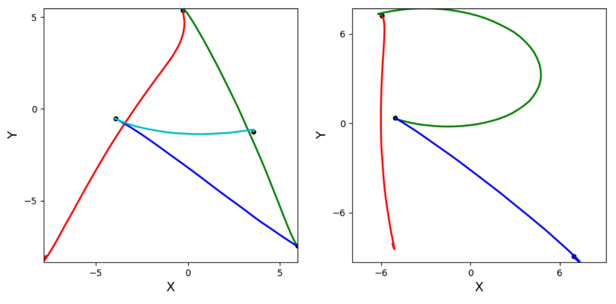
A. Segmentation Results

The results of the accuracy, precision, recall and F1-scores for segmentation are shown in Table I, plus a collection of pictures of the segmentation process in the letters dataset in Fig. 1. Also, each case of 3D Trajectories dataset are shown in Fig.2

Dataset	Precision [%]	Recall [%]	F1 [%]
2D Letters	99.24	100.00	99.62
3D Trajectories	81.35	97.96	88.89
Total Average	90.30	98.98	94.25

TABLE I: Segmentation Performance Results

The results indicate a good performance of the evaluated datasets. With an average precision of 90.30%, the model is correct in a high percentage of the proposed points. However, there is big difference between 2D and 3D datasets. While in 2D the algorithm is almost perfect with a 99.24% precision, these results indicate that there is excessive segmentation in the case of 3D dataset lowering the precision to 81.36%. The high average recall of 98.98% suggests that it effectively captures almost all of the desired segmentation points in both cases. The total average F1-score of 94.25% reflects a very strong result for the algorithm proposed, especially on 2D dataset but also good on the 3D. This suggests that the model demonstrates a satisfactory ability to correctly segment trajectories in various datasets. A future development would be to address the problem of over-segmentation in order to achieve an improvement in 3D precision.



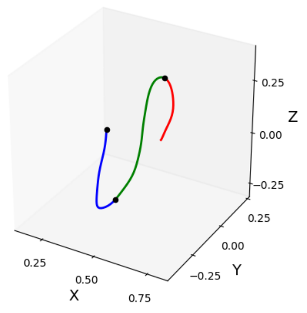
(a) Letter A (b) Letter R

Fig. 1: Example of representative 2D segmentation process for letters dataset.

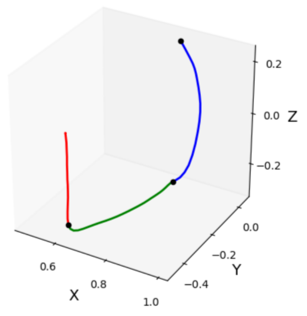
B. Sub-gesture Learning

The number of clusters has been defined empirically depending on the complexity and number of sub-gestures of each dataset. 10 clusters have been considered for the 2D dataset, and 14 for the 3D Trajectories dataset.

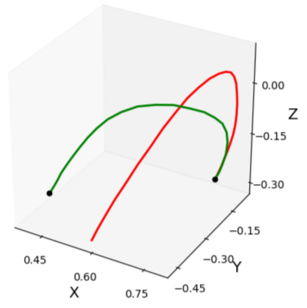
The accuracy table is shown in Fig. 3a. Also, an example of dictionaries for each dataset are shown in Tables 3b and 3c. These dictionary tables provide examples of the words associated with the various tasks, based on the concatenation of sub-gestures clustered into letters. In the first columns of the tables, the desired segmentation change-points are given as input but not the estimated ones, as we want to evaluate the sub-gesture learning stage without the possible errors on the previous stage. In the second columns, the given segmentation points



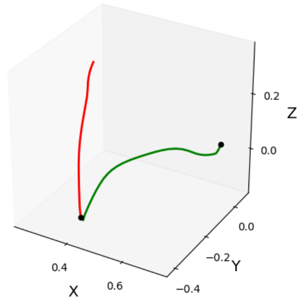
(a) S-shape Task



(b) U-shape Task



(c) Place-and-place Task



(d) Eating Soup Task

Fig. 2: Example of representative 3D segmentation process for 3D Trajectories dataset.

are the ones obtained by the algorithm. Hence, accuracy can be influenced not only on the performance of this section but on the previous segmentation accuracy.

Dataset	Accuracy [%] (Exp. Segm.)	Accuracy [%] (Alg. Segm.)
2D Letters	97.75	97.75
3D Trajectories	68.23	76.34
Total Average	82.99	87.05

(a) Accuracy Metrics

Task	Dict. Word (Exp. Segm.)	Dict. Word (Alg. Segm.)
A	hbcc	hbcc
E	jb	jb
K	abbcc	abbcc
M	abcb	abcb
R	abc	abc

(b) Letters Dataset dictionary considering first sample as validation set and remaining samples as training set

Task	Dict. Word (Exp. Segm.)	Dict. Word (Alg. Segm.)
S-shape	mbl	edl
U-shape	dfj	fbe
Place-and-place	gb	gac
Eating soup	ec	fb

(c) 3D Trajectories dictionary considering first sample as validation set and remaining samples as training set

Fig. 3: LOOCV sub-gesture learning performance results and example dictionaries

Accuracy metrics indicate a difference in performance between the 2D Letters and 3D Trajectories datasets. Regarding 2D Letters dataset, the high accuracy reflects a good model performance in identifying and classifying letter sub-gestures. In both expected and obtained segmentation cases, the accuracy is above 97%. Their similarity could be explained by the high precision on the segmentation process. In contrast, the performance has been lowered on 3D Trajectories dataset. The method has an accuracy of 68.24% when the theoretical perfect segmentation process is considered, and an accuracy of 76.34% when considering the segmentation points obtained by the algorithm. Essentially, it has been shown that the identification and classification of sub-gestures have been more difficult in this 3D trajectory dataset either with expected or algorithm obtained segmentation. Moreover, we can say that the inaccuracy of the learning is not caused by previous segmentation errors, since the results with the segmentation obtained by the algorithm are better than those obtained with the expected segmentation. To sum up, the overall average accuracy of 83% and 87.05% provides an overall acceptable perspective on both situations. However, there is room for improvement in the handling of 3D Trajectories.

C. *Gesture Classification*

We now look at the results of the comparison between the expected gestures with those obtained by the algorithm. As second columns again take segmentation obtained by the algorithm as input, these last columns are our final result and represent the accuracy of our whole method to classify a given gesture. However, this stage is strongly influenced by the prior learning of the sub-gesture. In fact, the classification accuracy would be perfect if all its sub-gestures were well classified. Therefore, what this section really evaluates is the ability of the algorithm to classify the total gesture well despite possible sub-gesture classification errors. In other words, the ability to correct classification errors of sub-gestures in the previous stage through classification of the entire gesture to its most probable case in the dictionary based on edit distance and the presented cluster-to-cluster and point-to-cluster similarities.

Dataset	Accuracy (Exp. Segm.)		Accuracy (Alg. Segm.)	
	C-C Sim.	P-C Sim.	C-C Sim.	P-C Sim.
2D Letters	96.00	96.00	96.00	96.00
3D Trajectories	68.42	73.68	73.68	81.58
Total Average	82.21	84.84	84.84	88.79

TABLE II: LOOCV gesture classification performance.

The results indicate a good performance of the gesture classification stage with an average accuracy that reaches 84% and 88% in the expected segmentation and algorithm situations respectively.

As we can see from these and previous tables, final accuracy is strongly affected by misclassification in sub-gesture learning. Nevertheless, the effectiveness of point-to-cluster similarity is shown to be valid. Unlike point-to-cluster similarity approach, point-to-cluster similarity has been able to correct some previous errors and increase the accuracy on 3D Trajectories. However, the effect on the 2D Letters Dataset is difficult to analyze since the initial accuracy was already very high, leaving few errors to correct.

Focusing on the results of the entire algorithm presented on two last columns of Tab. II, the accuracies are 96% and 81.59% for Letters and 3D Trajectories datasets respectively. Considering on the state-of-the-art results [17], the overall average accuracy of 88.79% can be considered as a satisfactory performance on gesture segmentation and recognition of trajectories. As seen on all the previous evaluation stages, the differences among datasets are significant. We show high performance on 2D charts, but an improvable result in 3D trajectories, which could be an area for future development.

From the whole analysis, we can see that a high accuracy is kept in the segmentation stage with an F1-score of 94.26%. Afterward, the first notable gap in the accuracy is generated reducing it to 87.05% due to the inaccuracy in gesture learning especially in 3D cases. Finally, accuracy is improved in the ultimate classification of gestures reaching a 88.79% thanks to NLP Levenshtein distance and point-to-cluster similarity.

As for the differences between the datasets, one possible cause of the drop on 3D sample could be the increase of variables in the characterization of the 3D trajectories.

VI. Conclusions

Gesture recognition is a useful tool for a wide variety of applications. In this paper we present a novel method for automatically segmenting and recognizing gestures generated by human-driven robots. Our method consists of two main steps: first, we decompose trajectories into sub-gestures, and second, we recognize and concatenate these sub-gestures to form complete gestures and classify them as several specific gestures. We evaluated our algorithm using two different datasets. After comparing our approach with the manual annotations of the surgical gestures using a LOOCV process, we observe an average F1-score of 94.25% on the segmentation, an accuracy of 87.05% on the learning task and an average accuracy of 88.79% for the fully automated gesture recognition. As shown, this method can satisfactorily segment and interpret new surgical gestures without human intervention. Future extensions can include the recognition of gestures within longer trajectories, to be able to compare it with other methods in literature such as [17].

Acknowledgments

This work was partially funded by CSIC Project 202350E080 ClothIRI: Robotic Cloth Manipulation at IRI. G. Tapia was also funded by JAE Intro 2023 Grant with reference EX1208.

References

[1] J. Rosen, B. Hannaford, C. G. Richards, and M. N. Sinanan, "Markov modeling of minimally invasive surgery based on tool/tissue interaction and force/torque signatures for evaluating surgical skills," *IEEE transactions on Biomedical Engineering*, vol. 48, no. 5, pp. 579–591, 2001.

[2] 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," *Computer Aided Surgery*, vol. 7, no. 1, pp. 49–61, 2002.

[3] N. Vaughan, B. Gabrys, and V. N. Dubey, "An overview of self-adaptive technologies within virtual reality training," *Computer Science Review*, vol. 22, pp. 65–87, 2016.

[4] F. Despinoy, D. Bouget, G. Forestier, C. Penet, N. Zemiti, P. Poignet, and P. Jannin, "Unsupervised trajectory segmentation for surgical gesture recognition in robotic training," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 6, pp. 1280–1291, 2015.

[5] S. Yang, T. S. Wells, R. A. MacLachlan, and C. N. Riviere, "Performance of a 6-degree-of-freedom active microsurgical manipulator in handheld tasks," in *2013 35th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pp. 5670–5673, IEEE, 2013.

[6] J. Casey, *Exploring curvature*. Springer Science & Business Media, 2012.

[7] Edelsbrunner, Letscher, and Zomorodian, "Topological persistence and simplification," *Discrete & computational geometry*, vol. 28, pp. 511–533, 2002.

[8] F. T. Pokorny, K. Goldberg, and D. Kragic, "Topological trajectory clustering with relative persistent homology," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16–23, IEEE, 2016.

[9] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Topological trajectory classification with filtrations of simplicial complexes and persistent homology," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 204–223, 2016.

[10] A. Colomé and C. Torras, "A topological extension of movement primitives for curvature modulation and sampling of robot motion," *Autonomous Robots*, vol. 45, no. 5, pp. 725–735, 2021.

[11] A. Tharwat, "Principal component analysis-a tutorial," *International Journal of Applied Pattern Recognition*, vol. 3, no. 3, pp. 197–240, 2016.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[13] S. Kullback and R. Leibler, "Letters to the editor," *The American Statistician*, vol. 41, no. 4, pp. 338–341, 1987.

[14] B. Varadarajan, C. Reiley, H. Lin, S. Khudanpur, and G. Hager, "Data-derived models for segmentation with application to surgical assessment and training," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2009: 12th International Conference, London, UK, September 20-24, 2009, Proceedings, Part I 12*, pp. 426–434, Springer, 2009.

[15] L. Tao, L. Zappella, G. D. Hager, and R. Vidal, "Surgical gesture segmentation and recognition," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2013: 16th International Conference, Nagoya, Japan, September 22-26, 2013, Proceedings, Part III 16*, pp. 339–346, Springer, 2013.

[16] H. C. Lin, I. Shafran, T. E. Murphy, A. M. Okamura, D. D. Yuh, and G. D. Hager, "Automatic detection and segmentation of robot-assisted surgical motions," in *International conference on medical image computing and computer-assisted intervention*, pp. 802–810, Springer, 2005.

[17] B. van Amsterdam, M. J. Clarkson, and D. Stoyanov, "Gesture recognition in robotic surgery: a review," *IEEE Transactions on Biomedical Engineering*, vol. 68, no. 6, 2021.