Clustering of Motion Trajectories by a Distance Measure Based on Semantic Features

Christoph Zelch[1], Jan Peters[2], Oskar von Stryk[1]

Abstract— **Clustering of motion trajectories is highly relevant for human-robot interactions as it allows the anticipation of human motions, fast reaction to those, as well as the recognition of explicit gestures. Further, it allows automated analysis of recorded motion data. Many clustering algorithms for trajectories build upon distance metrics that are based on pointwise Euclidean distances. However, our work indicates that focusing on salient characteristics is often sufficient. We present a novel distance measure for motion plans consisting of state and control trajectories that is based on a compressed representation built from their main features. This approach allows a flexible choice of feature classes relevant to the respective task. The distance measure is used in agglomerative hierarchical clustering. We compare our method with the widely used dynamic time warping algorithm on test sets of motion plans for the Furuta pendulum and the Manutec robot arm and on real-world data from a human motion dataset. The proposed method demonstrates slight advantages in clustering and strong advantages in runtime, especially for long trajectories.**

I. Introduction

Clustering of trajectories is an important tool for analysis, recognition and prediction of motions and has been used in humanoid robotics, e.g., to estimate the gait phase [1], to learn motion primitives from observations [2], to analyze posture data [3], or in the field of human-robot interactions to build a database structure for human motions that allows the robot to respond faster [4], to recognize hand gestures as input commands [5], [6], [7] or to recognize and predict human or robotic actions and movements [8], [9], [10].

Trajectories are typically represented as sequences of data points associated with time stamps. They can be extracted from image data, the output of a motion capture system or internal sensors that provide data for trajectories at discrete time steps. This data format can be interpreted as linear splines. Trajectory data may also originate from numerical planning or solutions of optimal control problems (for example, Merkt et al. [11]). Trajectories resulting from the latter can be represented as splines (e.g., in DIRCOL [12]), potentially cubic or even quintic splines to provide differentiability and smoothness. One application is the clustering of motion plans from different start states provided by the optimal control software DIRCOL [12] to differentiate between distinct solution clusters. In this work, we refer to motion plans as a combination of a state and a control trajectory. The distance measure is applicable to general trajectories as well as motion plans.

The insight that motivates this work is that the exact path of a trajectory is less relevant for clustering. Much more important is the sequence of some meaningful features (e.g., extrema, changepoints, roots, joint limits) and their salience in the trajectory. We thus do not focus on measuring small differences between very similar trajectories as this is, e.g., done when the error between a reference and some executed trajectory is computed. Instead, we are interested in a measure that captures differences in the general characteristic shape of trajectories to use this measure to cluster a set of trajectories into groups of similar graphs.

Contribution: In this work, we present a novel approach to transform robotic trajectory data into a sequence of high-level semantic features (which are characteristic points) describing the general shape of its graph. These features are supplemented with information about their temporal position and their salience in the trajectory. This feature sequence is used in an existing sub-sequence-based distance metric presented by Elzinga et al. [13], which we adapt for our application. The resulting distance measure for trajectories is used in agglomerative hierarchical clustering to identify trajectories with similarly shaped graphs. The advantage of this approach is that relevant feature classes can be chosen depending on the application or the considered problem. Our trajectory distance measure is compared with dynamic time warping (DTW) [14], we show that our approach outperforms this widely used measure on several test sets.

Related Work: Gaussian Mixture Models have been used by Lee [7], Piperakis et al. [1] or Luo et al. [9] for clustering trajectories. However, this approach requires the estimation of multiple Gaussian distributions which can be computationally costly and the result is sensitive to initial points. Research has been done to cluster trajectory data using (deep) neural networks (NN), e.g., using auto-encoders as in [1], [8], [15] or specialized structures like self-organizing incremental NNs [6]. NNs can provide good performance on large-scale datasets [15]. However, they require large amounts of training data to generalize well and provide limited or no insight into how the clustering is done.

Widely used clustering algorithms are hierarchical clustering (Basoeki et al. [3], Kulić et al. [2]), k-means (Maharani et al. [5]) or density-based clustering. The vast majority of clustering algorithms require a distance function to measure the distance or similarity of two trajectories. The construction of centroids needed for the k-means algorithm is not intuitive in the context of trajectory clustering. An advantage

[1]Christoph Zelch and Oskar von Stryk are with the Simulation, Systems, Optimization and Robotics Group, Department of Computer Science, TU Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany {zelch, stryk}@sim.tu-darmstadt.de

[2]Jan Peters is with the Intelligent Autonomous Systems Group, Department of Computer Science, TU Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany jan.peters@tu-darmstadt.de

of hierarchical over density-based clustering is the better interpretability of the results, as hierarchies of clusters can be represented in dendrograms. For these reasons, hierarchical clustering is used in this paper. Nevertheless, the proposed distance measure for trajectories can also be used combined with other distance-based clustering algorithms.

A large part of trajectory distance measures can be divided into those based on Lp-norms (well-known are, e.g., DTW and related, *Fréchet* distance) and based on edit distance (like *edit distance with real penalty*, *longest common subsequence*) [16]. Their properties have been studied extensively [17], [18]. These distance measures are typically applied to some condensed representation of the trajectories that are computed, e.g., using principal component analysis as in [1], multiple correspondence analysis [3] or hidden Markov models (HMM) [2], [6]. These reductions all are not intuitive to interpret, HMMs additionally require training.

In the context of social sciences, distance measures are often used to compare sequences of categorical elements from a finite alphabet. Distance measures for semantic sequences are reviewed in [19]. In this work, we build upon the work of Elzinga et al. [13], who present a measure called SVRspell, which we adapt to our problem. It is based on the number of matching sub-sequences in strings, which correspond to feature sequences of trajectories. The extraction of features from raw trajectories is related to Schmid et al. [20], who proposed a trajectory compression method for geographical movement data that uses semantic information to transform raw trajectory data. A review of this topic has been presented by Parent et al. [21]. However, there are several differences between the trajectory representations that reflect the specific requirements of our application. In particular, there is no underlying map; consequently, the raw trajectory cannot be reconstructed from our representation, in contrast to [20].

II. Description of the Method

Our clustering of a set of trajectories can be roughly divided into three major steps. In the first step, we reduce each trajectory to a sequence of high-level features augmented with additional information about timing and salience. In the second step, we use the sequence-based representation of the trajectories computed in step one to measure distances among them. The distances from the second step form a distance matrix used in the last step as input for standard agglomerative hierarchical clustering (see [22]) to identify similar trajectories in an examined set. These three steps will be detailed in the following subsections.

A. Definitions

In motion planning of robotic systems, there typically are state and control trajectories to describe the state of the system in combination with the control values required to generate the motion. In this paper, we call the combination of a state trajectory s and a control trajectory c synchronized in time *motion plan*:

$$T = \{s : [0, t_f] \to R_s \subset \mathbb{R}^n, c : [0, t_f] \to R_c \subset \mathbb{R}^m\}, \quad (1)$$

where $t_f > 0$ is the trajectory end time. We call trajectories with $t_f = 1$ time-normalized. In this work, we represent trajectories as mappings instead of the often-used sequence of values connected with time stamps since it is a better representation of the continuous reality than discrete sequences. Moreover, Equation (1) gives a more general representation, since these sequences can be interpreted as a continuous linear spline function.

Typical for motion planning for robotic systems is the existence of limits on the codomain of the trajectories that are induced for instance by joint limits or control bounds. In this work, we thus assume the existence of box-constraints on the codomains of the maps s and c, i.e.,

$$R_s := \prod_{i=1}^{n} [l_{s,i}, u_{s,i}] \quad (2)$$

for some lower and upper bounds $l_s, u_s \in \mathbb{R}^n$ and R_c analogously. Nevertheless, our approach is applicable with small changes to problems without bounds.

B. Step 1: Construction of Sequence-Based Representation

This subsection describes the extraction of features from time-dependent raw trajectory data. Following the definition of a multidimensional sequence in [23], we define a *feature sequence* as a sequence of elements $e_1, \dots, e_q$ where each element e_i is a triplet of attributes $e_i = \left(e_i^{\text{cat}}, e_i^{\text{time}}, e_i^{\text{val}}\right)$. In this triplet, e_i^{cat} denotes the kind of feature this element represents (*feature class*), e_i^{time} gives the time stamp of this feature in the normalized trajectory and $e_i^{\text{val}} \geq 0$ provides information about the importance of the feature in the sequence. Several examples of such attributes will be given later on. For a given trajectory, we compute $m+n$ sequences, one for each dimension in the state and control (cf. Eq. (1)).

The selection of relevant features is a crucial part of the application of this method and requires careful design. It depends on the one hand on the features that occur in at least some trajectories of the examined set and on the other hand on the relevance of these features for the problem or application underlying the trajectories. In this paper, we use maxima ($\wedge$) and minima ($\vee$), active box constraints (upper bound L_u, lower bound L_l) and for one problem roots (0) as feature classes. These are the dominant characteristics that are apparent in the trajectories' graphs resulting from our problems. Examples of other features are jumps, steep inclines or roots/signs of derivatives. The selection of relevant characteristics for the distance measure is highly problem specific.

Our state trajectories are represented as cubic splines, the control trajectory as linear splines. This is DIRCOL's output format, typical for collocation-based optimal control solvers. For different representations, it may be necessary to adapt the feature extraction approach. Trajectory data from sensors or images is typically a sequence of (potentially multidimensional) time-stamped values. This can be interpreted as linear splines, such that our approach for the control trajectories can be used for this data without further adaptation.

To extract features from a trajectory, we need for each feature class a proceeding to find the times e_i^{time} at which the

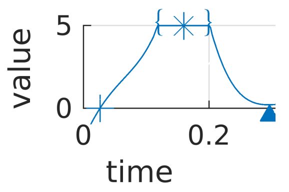


Fig. 1: The two events “start of constrained arc” ({) and “end of constrained arc” (}) and are merged into a single event “constrained arc” (∗) in the center of the constraint.

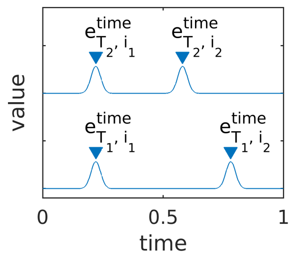
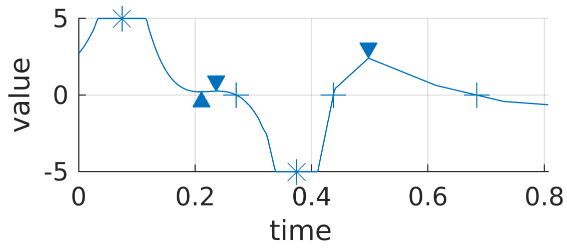


Fig. 2: Example of two simple trajectories, both have two features each, in different temporal distance.



(a) Trajectory graph with features indicated by symbols: ▲ for minima, ▼ for maxima, + for roots and ∗ for active constraints.

i	1	2	3	4	5	6	7	8
e^{cat}_i	L_u	$\vee$	$\wedge$	0	L_l	0	$\wedge$	0
e^{time}_i	0.07	0.21	0.24	0.27	0.37	0.44	0.50	0.68
e^{val}_i	0.23	0.01	0.01	0.97	0.74	1.00	0.19	0.93

(b) Feature sequence for the above trajectory graph. Feature labels: $\wedge$: maximum, $\vee$: minimum, 0: root, L_u/L_l: upper/lower box constraint active.

Fig. 3: Example showing the representation of a trajectory graph with a feature sequence.

feature e^{cat}_i occurs. Furthermore, we need to assign a value e^{val}_i to each feature representing its salience in the trajectory shape. The following describes our approach to get these features from linear and cubic splines.

1) Extrema: In linear splines, maxima and minima can only occur at knots (times at which the polynomial changes); we thus search for sign changes between the adjacent splines' slopes to find the extrema. For cubic splines, we search in each polynomial for extrema by checking if its derivative, a quadratic function, has roots inside the interval on which the spline is defined. In e^{cat}_i, we distinguish between *maximum* ($\wedge$) and *minimum* ($\vee$). To assign a value e^{val} to all maxima and minima, we use a concept called *(topographic) prominence* [24] as defined in [25] that is used, e.g., to value the significance of mountain summits. We slightly adapt this definition and normalize the computed prominence by dividing it by $u_s - l_s$ (or $u_c - l_c$). At this point, we use our assumption of bounded codomains. For unbounded codomains, one could still normalize the prominence using the inverse tangent or any other mapping $[0, \infty) \to [0, 1)$. Since prominence is defined for maxima only, we define the prominence of a minimum of a function f at t_m as the prominence of the maximum of the function $(-f)$ at t_m. Finally, we discard all extrema with a prominence lower than a certain threshold to avoid long feature sequences with many irrelevant features.

2) Active box constraints: To find the active box constraint feature in linear splines c, we again consider only knots at times t_i and check if the condition

$$u_c - c(t_i) < \varepsilon \text{ or } c(t_i) - l_c < \varepsilon \tag{3}$$

is fulfilled, with a small threshold $\varepsilon > 0$. For cubic splines, we also check the condition (3) at knots and extrema and proceed as for linear splines. Depending on whether the upper or lower bound is active, we use different attributes (L_u, L_l) in e^{cat}_i. Further, one could distinguish between a single touch point and the two points start and end of a constrained arc. In this work, we use a single feature for both cases and set the time e^{time}_i of an arc to the center point between the start and end of the active constraint (see Figure 1 for an example). To get salience values e^{val}_i, we treat active box constraints as extrema (which they actually are) and compute the prominence value at these points.

3) Roots: Considering roots can be helpful for motions where symmetry is relevant. The roots (e^{cat}_i indicated by 0) are identified by finding the roots of all (linear or cubic) polynomials that are part of the trajectory. The value of each root at e^{time}_i is computed using the slope $m = \frac{\mathrm{d}}{\mathrm{d}t} s(e^{\text{time}}_i)$ as

$$e^{\text{val}}_i = 2\pi^{-1} \left|\arctan(m)\right|. \tag{4}$$

Eq. 4 assigns salience values close to one to roots that intersect the x-axis with large slope and values close to zero for roots with flat angles. For all roots at knots of linear splines, we use the subderivative $m = \frac{1}{2}(m^+ + m^-)$, where m^- and m^+ denote the left and right limit, as they are not differentiable at these points. An example of a feature sequence for a one-dimensional trajectory, including time and salience values, is given in Figure 3.

C. Step 2: Distance Metric for Feature Sequences

For two motion plans T_1 and T_2, we construct the feature sequences as described in Section II-B. For these sequences, a distance needs to be computed to compare T_1 and T_2. We do this using the measure based on the string kernel method described in [13], [26], which allows incorporation of the features e^{time} and e^{val}. This measure is a metric, which means in particular that the triangular inequality holds, which “ensures coherence between computed dissimilarities” [19].

In the following, we briefly summarize Elzinga's distance measure *SVRspell* [13], [26]. Consider a finite alphabet Σ and the set Σ^* of all possible strings (character sequences) that can be built from Σ. Enumeration of the elements in Σ^* gives an infinite-dimensional vector space S where each entry corresponds to one string. For two strings $s_1, s_2 \in \Sigma^*$ the two corresponding vectors $r_{s1}, r_{s2} \in S$ list the number of respective sub-sequences that can be found in the strings. A short example is given in Table I.

To get the distance between s_1 and s_2, Elzinga et al. compute the Euclidean distance between r_{s1} and r_{s2}. This

$\Sigma = \{a, b\}$,	$\Sigma^* = \{a, b, aa, ab, ba, bb, aaa, \dots\}$
$s = abb$,	$r_s = \begin{pmatrix} 1 & 2 & 0 & 2 & 0 & 1 & 0 & \dots \end{pmatrix}$

TABLE I: Example of a string s and its vector r_s denoting how often all sub-sequences occur. In this example, the sub-sequences in s are *a*, *b*, *ab*, *bb*, *abb*.

can be formulated using only inner products, which can be computed even if the vectors are infinite-dimensional using so-called *kernels*:

$$\begin{aligned} d(s_1, s_2) &= \sqrt{\sum_{i \in \mathbb{N}} \left(r_{s1,i} - r_{s2,i}\right)^2} \qquad (5) \\ &= \sqrt{r_{s1}^T r_{s1} + r_{s2}^T r_{s2} - 2 r_{s1}^T r_{s2}}. \end{aligned}$$

In [13], several extensions of SVRspell are described, some of them are used in this work. In the following, we explain how we use this measure and its extensions to allow the application on our representation of motion plans from II-B.

The alphabet in [13] corresponds to our set of trajectory feature classes, a string with our feature sequence, and a character with a feature e_i^{cat}. We can compute the distance between motion plans by applying (5) to each dimension of our feature sequence representation of the state and control trajectories. The resulting $n + m$ distance values d_i are combined into a single distance measure by taking the root of the sum of squared distances:

$$d_{\text{final}} = \left(\sum\nolimits_{i=1}^{n+m} d_i\right)^{-\frac{1}{2}} \qquad (6)$$

The following extensions of [13] are used to include e^{time} and e^{val} in the distance measure.

Elzinga et al. describe how to add a concept named *soft-matching* to the string kernel, which allows quantification of similarity between distinct characters of the alphabet. As an example, consider the alphabet $\{a, b, c\}$ and the soft-matching given by the similarity 0.7 between a, b and 0.0 between both a, c and b, c. Then, the distance between words ab and aa is smaller than between the words ac and aa.
We make use of soft-matching to take into account the similarity of extrema and box constraints. The value that gives the similarity between $\wedge$ and L_u or $\vee$ and L_l is between 0 and 1, care must be taken that the resulting soft-matching matrix is positive definite.

The *Trail-algorithm* in [13] allows the introduction of a penalty term to decrease the influence of sub-sequences with large gaps. Gaps are differences between indices where a character of a sub-sequence occurs in a string. For example, the sub-sequence *abc* of the string *abac* has gaps 1 and 2 between *ab* and *ac*, respectively. This gap penalty can be an arbitrary data-dependent function, which allows us to incorporate the normalized trajectory times to penalize differences in the time spans between two features in their trajectories, as exemplified in Figure 2. To measure distances between feature sequences in our approach, we use the following gap weighting function:

$$g(i_1, j_1, i_2, j_2) = 1 - \left| \left(e^{\text{time}}_{T_1,i_2} - e^{\text{time}}_{T_1,i_1}\right) - \left(e^{\text{time}}_{T_2,j_2} - e^{\text{time}}_{T_2,j_1}\right) \right|. \qquad (7)$$

The inputs i_1 and i_2 are indices of two features in the trajectory T_1 and j_1, j_2 are indices for T_2. In the kernel, each entry of r_s of a feature sequence is weighted by a product of the gap weights computed by g in the respective sub-sequence (slightly simplified). How to handle the case where a sub-sequence occurs more than once in a string is described more detailed in [13].

Finally, we need to add the information provided by the salience e^{val} into the distance measure. In [13], Elzinga et al. propose an adaptation of their method to cover so-called *run-length encodings* of strings where repeated characters are encoded as numbers (e.g., *aaabccccb* as $a^3b^1c^4b^1$). This can also be used for "any quantifiable property of the characters", which is the salience e^{val} in our application. Accordingly, each sub-sequence of a string is weighted by the sum of such a run-length. However, products of the weights are more appropriate for our application. This ensures that the importance of sub-sequences containing features with very little salience is reduced as a whole. This extension requires a single line change in the existing algorithm: In the notion of [13], it suffices to modify line 5 of Elzinga's Grid-algorithm to assign $m^1_{ij} \leftarrow t_{xi} t_{yj}$.

All in all, *SVRspell* modified with three presented extensions (two of them already proposed in [13], one implemented for this work) provides a distance metric to compare feature sequences that represent motion plans. It allows incorporation of the additional information provided by temporal relation e^{time} and salience e^{val} of the features.

D. Step 3: Application of Hierarchical Clustering

The distance measure described in the previous section enables us to construct a distance matrix for a given set of motion plans, for which the distance between each pair in this set needs to be computed. This distance matrix is required by the agglomerative hierarchical clustering algorithm that constructs the set into subsets of similar motion plans. The merging of clusters is done using single-linkage. The number of resulting clusters is either pre-defined or results from a cutoff value. Both values must be tuned, which is facilitated by considering a dendrogram.

III. Evaluation and Experiments

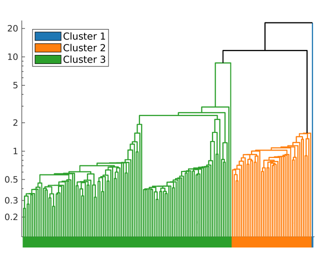
We compare the feature-based distance measure of motion plans based on SVRspell with DTW on the raw pairs of state and control trajectories. We selected DTW because it is, according to Su et al., "one of the most widely used distance measures" [16]. Computations are performed in Matlab R2020a on a laptop with an Intel i7-6500U processor with two cores at 2.5 GHz. We use the DTW implementation of the Matlab Signal Processing Toolbox (compiled code) and the agglomerative hierarchical clustering from the Matlab Statistics and Machine Learning Toolbox. The SVRspell algorithm is implemented in C++ and used in Matlab via MEX file. The feature extraction proposed in this paper is written in Matlab code. The sets of trajectories that are used in this evaluation are, if not stated otherwise, provided by the optimal control problem solver DIRCOL [12].

A. *Clustering of Furuta Pendulum Motion Plans*

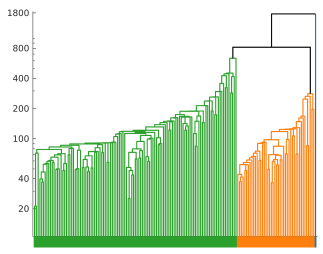
The underlying physical system of the motion plans in this subsection is the Furuta pendulum [27]. It is an underactuated rotary pendulum with two rotational joints, where the first joint can be actuated and the second joint rotates freely. An optimal control problem is formulated to get the controls for a swing-up motion that brings the passive second arm from a hanging start state into an upright position. The objective function favors solutions of minimum time and minimum energy. The problem formulation contains no explicit information about trajectory clusters or distance measures. Computing solutions of this optimal control problem from different start states sampled closely around the hanging state using the numerical solver DIRCOL gives a set of motion plans. The differences in the motion plans originate from the differences in the start states; the results provided by DIRCOL can be clustered into visually different solutions. By changing the parameterization of the dynamic model, we get three different test sets *Furuta 1* to *3* of 160 motion plans each that are clustered separately. We manually cluster the 160 trajectories for each test set and use this as ground truth. Trajectories with visually similar graphs are grouped into one cluster. The ground truth clustering has been done before developing our clustering approach.

For this problem, the *root* feature is used for the state trajectories as the roots of the second joint indicate the number of swing-up motions. A qualitative evaluation is performed by comparing the dendrograms and clusters resulting from DTW and our method. Each motion plan in the test set is listed on the x-axis of the dendrogram, the distances between clusters are on the y-axis. The trajectory plots show the distinct clusters of the motion plans, which contain state (consisting of two joint positions and two velocities) and control trajectories. The qualitative evaluation is supported by comparing the number of correctly clustered motion plans as quantitative evaluation. The separation into clusters depends on the cutoff value chosen. Lower cutoff values lead to fragmentation into more clusters. Misclassifications with high distance to the other elements make a lower cutoff value necessary to separate the clusters. The results for both methods are given for the separation into the most favorable number of clusters to allow a fair comparison. This means finding a reasonable tradeoff between the number of clusters and the lowest number of misclassifications.

The first test set *Furuta 1* has three different clusters containing 44, 115 and 1 trajectories. Both DTW and our method identify all clusters correctly (see Table II). The dendrograms (Fig. 4a, 4b) illustrate the subdivision into three clusters. The second test set *Furuta 2* has four different ground truth clusters with 90, 38, 1 and 31 trajectories. Our feature-based approach identifies the four clusters apart from two elements (Table II). DTW separates three single trajectories from the ground truth clusters and thus needs

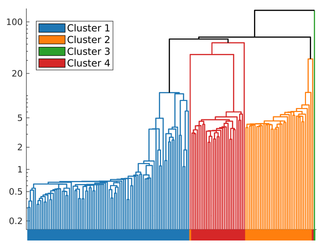


(a) Result with Feature-based

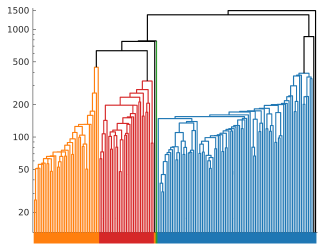


(b) Result with DTW

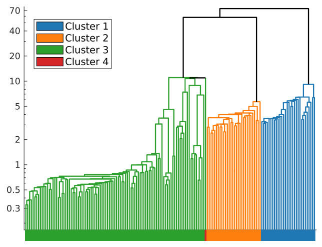
Fig. 4: The hierarchical structure of the clustering of test set *Furuta 1* using different distance measures. The ground truth is given as colored bar on the x-axis.



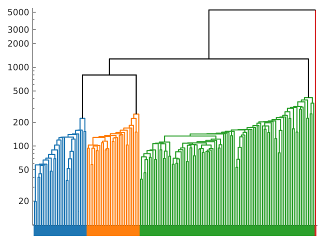
(a) *Furuta 2*: Result Feature-based



(b) *Furuta 2*: Result with DTW



(c) *Furuta 3*: Result Feature-based



(d) *Furuta 3*: Result with DTW

Fig. 5: The hierarchical structure of the clustering of test sets *Furuta 2* and *Furuta 3* using different distance measures.

a lower cutoff value, leading to seven clusters. Supporting the quantitative result, the dendrogram for the feature-based approach is structured more clearly: The four clusters are separated and the more fine-grained splitting happens only at a lower cutoff level.

The third test set *Furuta 3* consists of four trajectory clusters with sizes 30, 30, 99 and 1. On this test set, the clustering computed with DTW equals the manual clustering, the feature-based distance measure assigns the single trajectory cluster to the largest cluster.

B. *Clustering of Manutec r3 Arm Motion Plans*

In this subsection, we consider a more complex robotic system. The Manutec r3 robot arm model has three actuated joints, such that a motion plan consists of a six-dimensional state and a three-dimensional control trajectory. The motion plans that describe optimal point-to-point movements from different start states to the same goal state are computed again using the solver DIRCOL. The test set based on the Manutec arm consists of 30 motion plans that are clustered

GT	*Furuta 1*						*Furuta 2*									*Furuta 3*						
	Feature-based (3)			DTW (3 clusters)			Feature-based (4)				DTW (7 clusters)					Feature-based (3)			DTW (4 clusters)			
	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4	…	C_1	C_2	C_3	C_1	C_2	C_3	C_4
C_1	**1**			**1**			**90**				**88**				2	**30**			**30**			
C_2		**44**			**44**			**37**		1		**37**			1		**30**			**30**		
C_3			**115**			**115**			**1**				**1**					**99**			**99**	
C_4	–	–	–	–	–	–		1		**30**				**31**				1				**1**

TABLE II: The clusterings of the Furuta test sets as confusion matrices using the feature-based approach and DTW. The rows give the ground truth clusters, the columns the respective cluster assignments resulting from the two distance measures. The number of clusters is given as number behind the method name. Additional clusters are summarized in "…".

manually into 9 different movements. There are three larger clusters of 10, 6 and 5 motions, respectively. The other clusters contain one or two trajectories and are neglected in this evaluation.

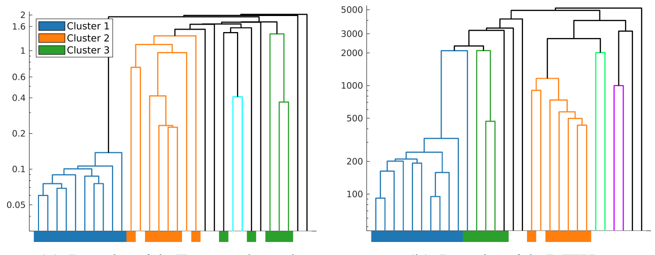
A property of motion plans resulting from this problem is a low-amplitude zig-zag behavior in one dimension of the control trajectory. This validates our use of a small prominence filter (a salience threshold of 0.02 is sufficient) for the extrema to remove the many maxima and minima. The prominence filter is thus important to reduce the length of the feature sequence. Moreover, it can compensate for some noise in the input data. The feature class *root* provides no useful information for trajectories of the Manutec test set and is thus not used for the state or control trajectories.

Table III shows that, considering the number of correctly clustered motion plans in the three largest clusters, our feature-based approach has a slight advantage over DTW: The large cluster 1 is identified correctly, but cluster 3 is incomplete and the algorithm has problems separating the second cluster from other motion plans. In contrast, DTW is unable to successfully identify any cluster: The second cluster includes one additional motion plan, the first cluster contains four from cluster 3. Comparing the dendrograms (Fig. 6), the diagram for DTW seems to be more clearly structured. The first cluster is presented contiguously, but a very low cutoff value is needed to separate it from other elements. In the dendrogram for our approach, this cluster is also clearly recognizable: It is separated from other motion plans, even for a very high cutoff value. Cluster 1 stands out visually from the other motion plans, it is clearly identified by the feature-based approach, whereas DTW has major problems with this cluster, as can be seen in the dendrogram.

All in all, both methods have some difficulties with the selected data set, which is reflected in the dendrograms. In several cases, the motion plans have similar progressions in single trajectories, which makes clustering challenging even for human experts. Interestingly, the two approaches identify different clusters well. However, considering the overall number of correctly classified motion plans, our feature-based approach has a slight edge on DTW.

C. Clustering of a Real-world Human Motion Dataset

To evaluate the performance of the presented method on real-world data, we cluster trajectories from the human locomotion dataset created by Reznick et al. [28]. They



(a) Result with Feature-based (b) Result with DTW

Fig. 6: The hierarchical structure of the clustering of test set *Manutec* using different distance measures.

GT	Feature-based (11)				DTW (10)			
	C_1	C_2	C_3	…	C_1	C_2	C_3	…
C_1	**10**				**10**			
C_2		**6**				**6**		
C_3			**3**	2	4		**1**	
…		3		…		1		…

TABLE III: The results of the clustering as confusion matrix for the Manutec robot arm test set comparing feature-based and DTW distance measures. Only the three largest clusters are considered, all others are summarized in "…".

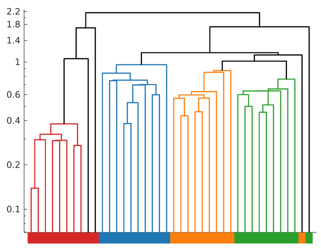
recorded data from ten participants walking, running, stair climbing and sitting down/standing up at various speeds and inclinations. The recorded data is post-processed (filtered and gaps in data filled). We use the normalized dataset where the strides in the recorded trajectories are separated and the time is normalized to 1 and interpolated to 150 data points (for details, see [28]). In our evaluation, we use the provided joint angle trajectories for the ankle, knee, hip, pelvis and foot progression, and from each of them only the x-component which is the most expressive and thus provides the best results. Consequently, the state trajectories to be clustered have five dimensions. There is no control signal information, the joint limits are not known and roots have no meaning, such that we use only the state trajectory with maxima and minima as features. Extrema with prominence lower than 0.2 are removed from the feature sequence.

A cluster consists of the first stride of all ten participants in a single motion task. The distinctness of the different motions in the data set varies, such that the difficulty of the clustering task varies with the selection of motions. For example, differentiating between joint trajectories for walking

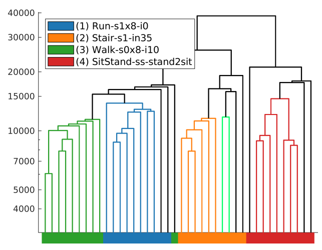
GT	Human Motion 1 Feature-based (8)					DTW (13 clusters)					DTW (4 clusters)			
	C_1	C_2	C_3	C_4	...	C_1	C_2	C_3	C_4	...	C_1	C_2	C_3	C_4
C_1	**10**					**8**				2	**0**		10	
C_2		**9**			1		**6**			4		**10**		
C_3			**9**		1			**9**		1			**10**	
C_4				**8**	2				**7**	3	1			9

GT	Human Motion 2 Feature-bd. (13)					Feature-bd. (10)					DTW (13)					DTW (8)				
	C_1	C_2	C_3	C_4	...	C_1	C_2	C_3	C_4	...	C_1	C_2	C_3	C_4	...	C_1	C_2	C_3	C_4	...
C_1	**8**				2	**9**				1	**7**		2		1	**10**				
C_2		**10**					**10**					**8**			2	1	**8**			1
C_3	1		**4**		5	5	1	**1**		3			**8**		2	9		**1**		
C_4				**8**	2				**8**	2				**4**	6				**7**	3

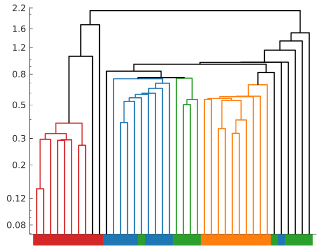
TABLE IV: Clustering results as confusion matrices for the Human motion dataset [28]. Motion types (ground truth) given in rows, assigned clusters in columns. Two separations into clusters are given if it is unclear which one is best.



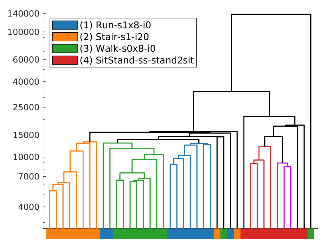
(a) *Human Motion 1*: Result with Feature-based approach



(b) *Human Motion 1*: Result DTW



(c) *Human Motion 2*: Result with Feature-based approach



(d) *Human Motion 2*: Result DTW

Fig. 7: The hierarchical structure of the clustering of test sets *Human Motion 1* and *Human Motion 2* using different distance measures. Colored bars represent ground truth.

with different inclinations is also difficult for humans; both clustering methods fail to cluster these trajectory bundles. In the test set *Human Motion 1*, we cluster into the following motion types which are better to differentiate: (1) running at $1.8\,\mathrm{m/s}$, (2) descending a $35°$ stair, (3) walking $10°$ downhill at $0.8\,\mathrm{m/s}$ and (4) sitting down on a chair. The results of the clustering using DTW and the feature-based approach are presented in Fig. 7a–7b. By and large, all four clusters can be identified in the dendrograms. The feature-based approach has a problem to distinguish between the clusters (2) and (3), DTW to differentiate between (1) and (3). Both approaches report high distance values for some trajectories of a motion type and thus separate them from the others, resulting in more than four clusters. As before, we report in Table IV (upper) the separation into the most favorable number of clusters for the feature-based approach and for DTW two alternative separations into 13 and 4 clusters. It can be seen that the four ground truth motions are identified with much higher cutoff values using the feature-based approach, and also the number of misclassifications is substantially lower.

More difficult to differentiate are the motion types of *Human Motion 2*: (1) running at $1.8\,\mathrm{m/s}$, (2) ascending a $20°$ stair, (3) walking at $0.8\,\mathrm{m/s}$ and (4) sitting down on a chair. The results are presented in Fig. 7c–7d. Here, the ground truth clusters are more difficult to distinguish in both dendrograms. The feature-based approach needs a separation into 13 clusters to distinguish four larger clusters, DTW also needs 13 clusters to distinguish between cluster (1) and (3). The results for different numbers of clusters are given in Table IV (lower). All in all, the performance of both approaches is comparable on this more difficult test set; both reveal difficulties in differentiating between cluster (1) and cluster (3) in particular.

D. Evaluation of Efforts and Runtime

The runtime analysis of the feature-based distance computation can be divided into the feature extraction step and the *SVRspell*-based distance computation. The feature extraction is a preprocessing step that must be done only once for all trajectories, independent of the number of distance measurements required for the clustering. Its runtime depends on the number of feature classes and the algorithms employed to identify the features of a trajectory. The complexity of the distance computation (based on the SVRspell algorithm) is cubic [13] in the *number of features*. The number of features is typically much smaller than the number of time steps, on which the complexity of DTW depends ($\mathcal{O}(NM)$ with N, M the trajectories' number of time steps).

The runtime of the *SVRspell*-based distance computation is important, as the number of calls to the distance computation is quadratic or even cubic in the number of trajectories for many clustering approaches [22]. Fig. 8 shows the time to extract the feature sequence (orange) as well as the times to compute the distance between two trajectories using DTW (blue) and feature-based approach (red). It must be noted that for this evaluation, the DTW and SVRspell implementations run as compiled code while the feature extraction is implemented in plain Matlab. The distance measure based on the feature sequence representation has clear advantages regarding computation time.

IV. Discussion, Conclusion and Outlook

The number of features has the largest impact on the runtime complexity of the new distance measure, which effectively decouples the computation of the distance from the time length of a trajectory. This is an advantage if long trajectories can be reduced to a small number of features. The number of features per trajectory depends on the trajectory itself, the feature classes used and the parameterization

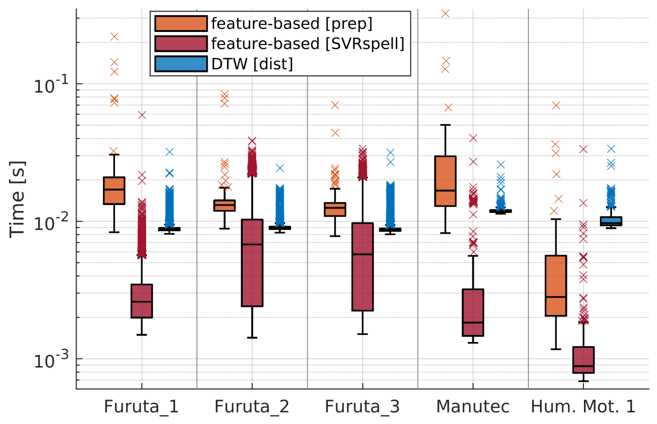


Fig. 8: Computation time for a single distance computation between two trajectories on the test sets considered here. Computation time of the feature-based approach is splitted in precomputation step and *SVRspell* distance computation.

(thresholds). An advantage of our method is flexibility in the selection of arbitrary many and arbitrary complex feature classes, which allows adaptation to specific problems and use cases. The parameters of the presented method are feature class dependent; typical parameters are thresholds below which features are considered as irrelevant and are discarded to keep the feature sequence as small as possible. It must be noted that our distance measure needs a sufficient number of distinct features in the trajectory graph to show its potential.

We have proposed a new approach to cluster trajectories that represent motion plans. It is based on a method [26] to measure the distance between strings. We present an approach to compress trajectories to sequences of user-defined feature classes to make it applicable for trajectories. Overall, the results show a reliable performance and several advantages compared to the proven DTW. In some cases, it outperforms this widely used distance measure.

In the field of human-robot interactions, further potential of the presented distance measure lies in the construction of hierarchical motion databases (as done, e.g., in [2], [4]), since the compressed feature-based representation of motions is memory efficient and allows fast comparisons.

References

[1] S. Piperakis, S. Timotheatos, and P. Trahanias, "Unsupervised gait phase estimation for humanoid robot walking," in *2019 Int. Conf. on Robotics and Automat. (ICRA)*. IEEE, 2019, pp. 270–276.

[2] D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura, "Incremental learning of full body motion primitives and their sequencing through human motion observation," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330–345, 2012.

[3] F. Basoeki, F. Dalla Libera, E. Menegatti, E. Pagello, and H. Ishiguro, "Clustering of humanoid robot motions executed in response to touch," in *Intelligent Autonomous Systems 13: Proceedings of the 13th International Conference IAS-13*. Springer, 2016, pp. 1063–1076.

[4] K. Yamane, M. Revfi, and T. Asfour, "Synthesizing object receiving motions of humanoid robots with human motion database," in *2013 IEEE Int. Conf. on Robot. and Automat.* IEEE, 2013, pp. 1629–1636.

[5] D. A. Maharani, H. Fakhrurroja, R. Machbub, and C. Machbub, "Hand gesture recognition using k-means clustering and support vector machine," in *2018 IEEE symposium on computer applications & industrial electronics (ISCAIE)*. IEEE, 2018, pp. 1–6.

[6] S. Okada, Y. Kobayashi, S. Ishibashi, and T. Nishida, "Incremental learning of gestures for human–robot interaction," *AI & society*, vol. 25, pp. 155–168, 2010.

[7] S.-W. Lee, "Automatic gesture recognition for intelligent human-robot interaction," in *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*. IEEE, 2006, pp. 645–650.

[8] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, "Trajectory clustering via deep representation learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3880–3887.

[9] R. Luo, R. Hayne, and D. Berenson, "Unsupervised early prediction of human reaching for human–robot collaboration in shared workspaces," *Autonomous Robots*, vol. 42, pp. 631–648, 2018.

[10] C. Sung, D. Feldman, and D. Rus, "Trajectory clustering for motion prediction," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1547–1552.

[11] W. X. Merkt, V. Ivan, T. Dinev, I. Havoutis, and S. Vijayakumar, "Memory clustering using persistent homology for multimodality-and discontinuity-sensitive learning of optimal control warm-starts," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1649–1660, 2021.

[12] O. von Stryk, "Numerical solution of optimal control problems by direct collocation," in *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*. Birkhäuser Basel, 1993, vol. 111, pp. 129–143.

[13] C. H. Elzinga and H. Wang, "Versatile string kernels," *Theoretical Computer Science*, vol. 495, pp. 50–65, 2013.

[14] N. Vaughan and B. Gabrys, "Comparing and combining time series trajectories using dynamic time warping," *Procedia Computer Science*, vol. 96, pp. 465–474, 2016.

[15] X. Olive, L. Basora, B. Viry, and R. Alligier, "Deep trajectory clustering with autoencoders," in *ICRAT 2020, 9th International Conference for Research in Air Transportation*, 2020.

[16] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, "A survey of trajectory distance measures and performance evaluation," *The VLDB Journal*, vol. 29, pp. 3–32, 2020.

[17] Y. Tao, A. Both, R. I. Silveira, K. Buchin, S. Sijben, R. S. Purves, P. Laube, D. Peng, K. Toohey, and M. Duckham, "A comparative analysis of trajectory similarity measures," *GIScience & Remote Sensing*, vol. 58, no. 5, pp. 643–669, 2021.

[18] M. Vagni, N. Giordano, G. Balestra, and S. Rosati, "Comparison of different similarity measures in hierarchical clustering," in *2021 IEEE Int. Symp. on Med. Meas. and Appl. (MeMeA)*, 2021, pp. 1–6.

[19] M. Studer and G. Ritschard, "What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures," *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, pp. 481–511, 2016.

[20] F. Schmid, K.-F. Richter, and P. Laube, "Semantic trajectory compression," in *Advances in Spatial and Temporal Databases*. Springer, 2009, pp. 411–416.

[21] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, and Z. Yan, "Semantic trajectories modeling and analysis," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, pp. 1–32, 2013.

[22] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview, ii," *WIREs Data Mining and Knowledge Discovery*, vol. 7, no. 6, p. e1219, 2017.

[23] A. S. Furtado, D. Kopanaki, L. O. Alvares, and V. Bogorny, "Multidimensional similarity measuring for semantic trajectories," *Transactions in GIS*, vol. 20, no. 2, pp. 280–298, 2016.

[24] M. Llobera, "Building past landscape perception with gis: Understanding topographic prominence," *Journal of Archaeological Science*, vol. 28, no. 9, pp. 1005–1014, 2001.

[25] A. Kirmse and J. de Ferranti, "Calculating the prominence and isolation of every mountain in the world," *Progress in Physical Geography*, vol. 41, no. 6, pp. 788–802, 2017.

[26] C. H. Elzinga, "Sequence similarity: A nonaligning technique," *Sociological Methods & Research*, vol. 32, no. 1, pp. 3–29, 2003.

[27] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing-up control of inverted pendulum using pseudo-state feedback," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 206, no. 4, pp. 263–269, 1992.

[28] E. Reznick, K. R. Embry, R. Neuman, E. Bolívar-Nieto, N. P. Fey, and R. D. Gregg, "Lower-limb kinematics and kinetics during continuously varying human locomotion," *Scientific Data*, vol. 8, no. 1, p. 282, 2021.