Trajectory Classification in n Dimensions using Subspace Projection

Thomas Nierhoff - Sandra Hirche

Abstract— **This paper presents a novel descriptor for trajectory classification in n dimensions, which is invariant with respect to scaling and rigid transformation. Using a hierarchical approach, the descriptor is able to capture both local and global features of the trajectory. The algorithm iteratively splits up every trajectory into smaller trajectory segments resulting in a binary tree. Inspired by the Frenet-Serret formulas, a projection onto a lower dimensional subspace is performed for every trajectory segment, providing a characteristic description of every trajectory. The subspace projection acts as a pseudo-curvature measure in every dimension. Successful applicability is shown through classification experiments in three and six dimensions using an RGB-D camera. For comparison with other algorithms, the Australian Sign Language dataset is also used for classification, showing a superior classification rate.**

I. Introduction

Tracking and classifying human movements is an important issue for cognitive human-robot interaction if a robot is supposed to react properly to a human gesture or movement. If the classification method of the trajectory, e.g. of the hand, can be spatially invariant but temporal-variant, Hidden Markov Models or Support Vector Machines are suitable approaches for classification. Problems occur if the trajectory to be recognized is allowed to vary also in size, orientation, and position. Depending on the number of dimensions, this involves different tasks in various fields of robotics and computer science: One representative problem in three dimensions is the distinction between human activities focusing either on whole body movements or only single limbs as demonstrated in [1]. In two dimensions, the problem includes handwriting classification and video surveillance tracking pedestrians or cars [2]. Higher-dimensional problems include the combination of a 3D position, a 3D orientation and a 6D wrench, as investigated in [3].

When classifying trajectories, one can distinguish between several substantially different approaches depending on the inherited invariance of the trajectory classifier. Without any invariant descriptor, trajectories have to be identical in the spatial and temporal domain for being matched perfectly. For a temporal invariant description of trajectories, Dynamic Time Warping as described in [4] is nowadays a well established method. For a rigid transformation invariant trajectory description, different approaches do exist. In [5] a force-field method is presented tackling this problem for the three-dimensional case. Another method working via pose normalization by capturing the global shape of the trajectory is PCA, see [6], [7]. Curvature-based methods on the other hand, measuring the torsion and curvature of a trajectory segment, are able to focus also on local trajectory properties. Two approaches measuring local trajectory properties are presented in [8], [9]. Another way for categorizing descriptors is the distinction between hierarchical and non-hierarchical descriptors. Non-hierarchical descriptors work at a certain granularity level of the trajectory usually focusing on the local features of the trajectory, see [10]. Hierarchical descriptors on the other hand begin with a coarse description of the trajectory but iteratively split it up into smaller trajectory segments, thus capturing both global and local properties of the trajectory well. Approaches based on hierarchical descriptors are introduced in [11] and [12]. In [13] a projection approach is used for classification of trajectories, which inspired our work presented in this paper. A common drawback of all presented hierarchical descriptors so far is that none of them is spatially invariant with respect to changes in orientation, translation and size and at the same time able to work in an arbitrary large number of dimensions.

The contribution of this paper is the development of an spatially invariant hierarchical descriptor for trajectory classification working in any number of dimensions. For a n-dimensional trajectory, a set of subspaces of different lower dimensions is created. By combining this idea with an hierarchical approach, one can capture both global and local properties of the trajectory. For this, each trajectory (segment) is divided iteratively into two smaller trajectory segments. This results in a binary tree of trajectory segments for which subspace projection is applied. The resulting subspace projection distances in combination with additional supplementary measures (PCA eigenvalues, subspace condition number) are compared for different binary trees forming the final similarity measure. The resulting descriptor is scaling and rigid-body transformation invariant. Experimental studies are based on a dataset with up to 40 different classes of motions with 5 trajectories for each class recorded by using an RGB-D camera. Results show that the algorithm performs well under various conditions with classification rates up to 94%. Comparison with other descriptors based on the Australian Sign Language dataset show superior classification rates of the proposed approach in this paper.

The remainder of this paper is organized as follows: Sec. II illustrates the general approach of the proposed descriptor. In Sec. III the feasibility is evaluated through different classification scenarios. Sec. III-C contains a critical analysis of the algorithm, discussing both its potential and limitations.

Thomas Nierhoff and Sandra Hirche are with the Institute of Automatic Control Engineering (LSR), Faculty of Electrical Engineering, Technische Universität München, D-80290 München, Germany {tn, hirche}@tum.de.

II. Classifier Determination

This section describes the three-staged approach for comparing two trajectories. Similar to [14] first a hierarchical description of the trajectory called *shape-tree* is created in order to represent the trajectory at different granularity. The generated *curvature-tree* then covers all important information about the trajectory using a combination of subspace projection and PCA. By comparing *curvature-trees* of different trajectories, the final similarity measure is obtained.

A. Shape-Tree

In this section the splitting process is described which will return a binary tree called *shape-tree* facilitating the comparison of different trajectories. The motivation to construct the *shape-tree* is to find a simplified description of every trajectory at different detail levels. Consequently, the topmost node (root) of the *shape-tree* provides only with a rough idea of what the trajectory will look like whereas the bottommost level represents an encoding of the entire trajectory.

Let $\mathcal{T}_{0p}$ denote a trajectory consisting of an ordered set of sampling points $(\mathbf{t}_0, \ldots, \mathbf{t}_p) \in \mathbb{R}^n$. For the splitting process one has to find a sampling point $\mathbf{t}_i \in \mathcal{T}_{0p}$ such that the two resulting lines through $(\mathbf{t}_0, \mathbf{t}_i)$ and $(\mathbf{t}_i, \mathbf{t}_p)$ minimize a certain similarity measure. For this purpose, the original trajectory is split up into two subtrajectories $\mathcal{T}_{0i} = (\mathbf{t}_0, \ldots, \mathbf{t}_i)$ and $\mathcal{T}_{ip} = (\mathbf{t}_i, \ldots, \mathbf{t}_p)$. Then the first line $\mathcal{L}_{0i}$ through $(\mathbf{t}_0, \mathbf{t}_i)$ matches $\mathcal{T}_{0i}$ best and the one second line $\mathcal{L}_{ip}$ through $(\mathbf{t}_i, \mathbf{t}_p)$ matches $\mathcal{T}_{ip}$ best. Inspired by surface energy considerations the following approach is proposed: Let $\mathbf{t}_k^{proj}$ be the orthogonal projection of a sampling point $\mathbf{t}_k \in \mathcal{T}_{0i}$ on $\mathcal{L}_{0i}$. Then the weighted squared length L_{0i} of the trajectory and the sum of squares S_{0i} is calculated as

$$L_{0i} = \quad w_L \left(\sum_{k=0}^{i-1} \| \mathbf{t}_{k+1} - \mathbf{t}_k, \| \right)^2, \tag{1}$$

$$S_{0i} = \quad \sum_{k=0}^{i} \left\| \mathbf{t}_k - \mathbf{t}_k^{proj} \right\|^2, \tag{2}$$

with w_L as a weighting factor counting the number of sampling points of the trajectory segment. The same calculation is done for any point $\mathbf{t}_k \in \mathcal{T}_{ip}$ and the second line, resulting in S_{ip} and L_{ip}. The optimal sampling point $\mathbf{t}_s$, named *splitting point*, splitting the trajectory into two subtrajectories is

$$\mathbf{t}_s = \underset{t_i \in \mathcal{T}_{0p}}{\operatorname{argmin}} (L_{0i} + S_{0i} + L_{ip} + S_{ip}), \; i = 0, \ldots, p. \tag{3}$$

The values S_{0i} (resp. S_{ip}) and L_{0i} (resp. L_{ip}) act as energy terms stored in the two different types of concurrent "mechanical springs": Whereas minimizing S_{0i} and S_{ip} means to match the trajectory as close as possible to the two line segments in the least-squares sense (constrained minimal surface problem), minimizing L_{0i} and L_{ip} centers the *splitting point* with respect to the arc length of the trajectory segment in the case of equidistant sampling points. The factor w weights L_{0i} and L_{ip} with respect to S_{0i} and S_{ip} in a suitable way.

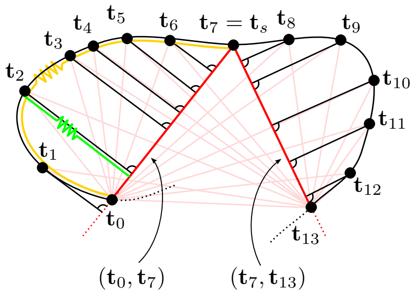


Fig. 1. Determination of $\mathbf{t}_s$ for the two-dimensional case. A green spring for $\left\| \mathbf{t}_2 - \mathbf{t}_2^{proj} \right\|$ and an orange spring for L_{07} represents the analogy in terms of an energy minimization.

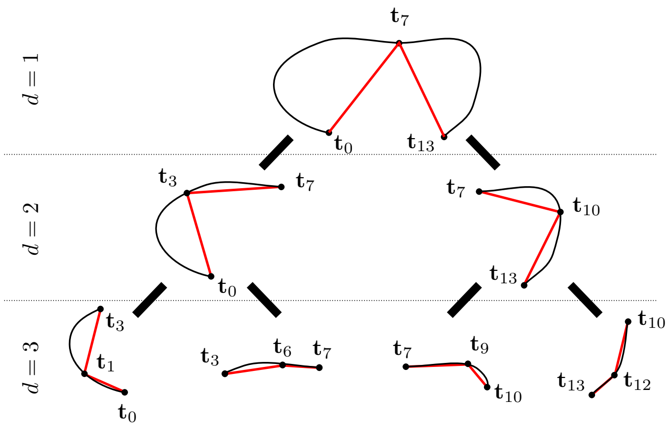


Fig. 2. Creation of the the *shape-tree*. Each found splitting point divides a trajectory segment into two subtrajectories. Furthermore, a *depth*-value shown on the left side is assigned to each object in the shape-tree.

Each trajectory segment - the overall trajectory and all splitted subtrajectories - consist of a *start point* denoting the chronological first point and an *end point* as the chronological last point of the trajectory segment. Iteratively one can continue with this procedure until each subtrajectory consists only of two subsequent sampling points $(\mathbf{t}_{j-1}, \mathbf{t}_j)$, $j \in (1, \ldots, p)$. If a trajectory segment cannot be split up anymore *depth*-value $d_{cur} < d_{max}$, it will have only one subtrajectory which is identical to the trajectory segment. The resulting structure can be stored in a binary tree, called *shape-tree*. Starting from 1 and ranging till d_{max} on for the original trajectory, a *depth*-value d is assigned for each subtrajectory depending on its position in the binary tree.

B. Curvature-Tree

After having created the *shape-tree*, a similarity measure between different trajectories needs to be established. For this purpose, a binary tree called *curvature-tree* $\mathcal{C}$ with similar structure to the *shape-tree* is created. Stored in each node of the *curvature-tree* are a set of measures explained in this subsection. In the style of the Frenet-Serret formulas, the distance between a point and its projection onto a lower-dimensional subspace for each trajectory segment of the *shape-tree* is used as a first measure. It is defined as

$$S_q = \text{span}(\mathbf{t}_{s2} - \mathbf{t}_{s1}, \dots, \mathbf{t}_{sq} - \mathbf{t}_{s1}), \quad q = 2, \dots, n, \tag{4}$$
$$\mathbf{t}_s - \mathbf{t}_{s1} = \mathbf{t}_{\parallel q} + \mathbf{t}_{\perp q}, \quad \mathbf{t}_{\parallel q} \in S_q, \mathbf{t}_{\perp q} \perp S_q, \tag{5}$$
$$l_{\perp q} = \|\mathbf{t}_{\perp q}\|, \tag{6}$$
$$\tag{7}$$

with $\mathbf{t}_s$ being the *splitting point*, $\mathbf{t}_{s1}$ the *start point* and $\mathbf{t}_{s2}$ the *end point* of the trajectory segment. In case of a n-dimensional space first a set of bases S_q determining subspaces with dimension $1, \dots, n-1$ are defined. Then the vector $\mathbf{t}_s - \mathbf{t}_{s1}$ is split up into a normal component $\mathbf{t}_{\parallel q} \in S_q$ and an orthogonal component $\mathbf{t}_{\perp q} \perp S_q$. The value $l_{\perp q}$ forms the principal component for comparing trajectories as described in the next section.

For $n > 2$, additional sampling points beside $\mathbf{t}_{s2}$ and $\mathbf{t}_{s1}$ as stated in Eq. (4) have to be defined in order to form a basis for the subspaces S_q. Two methods are suggested: For the *parent method*, new sampling points are selected by processing the *shape-tree* towards its root. On the other hand, the *neighbor method* adds new sampling points by processing the adjacent nodes of the same *depth*.

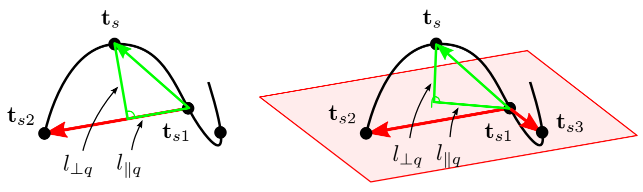


Fig. 3. Subspace projection for a 1D subspace (left side) and a 2D subspace (right side).

A problem occurs if the basis for S_q is ill-conditioned, see Fig. 4. This effect may happen either accidentally if two sampling points forming an basis vector are very close to each other or if the subspace projection is degenerated, e.g. if the entire trajectory lies on a lower dimensional subspace. To measure the effect, the condition number based on a SVD of the basis vectors of S_q is calculated. Because it is unknown whether a basis is ill-conditioned by accident or due to degeneracy, the eigenvalues $\lambda_e, \ e = 1, \dots, n$ of the PCA of all sampling points of the trajectory segment are calculated as well. All values - subspace projection length,

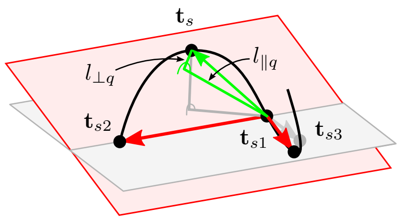


Fig. 4. Effect of an ill-conditioned basis. Moving $\mathbf{t}_{s3}$ only slightly along the trajectory (grey vs. black dot) causes the length of the projection vector $l_{\perp q}$ to change heavily.

condition number and PCA eigenvalues form the vector of scalar values stored in each node of the *curvature-tree*.

C. Curvature-Tree Comparison

Using the scalar values stored in the *curvature-tree* to measure the similarity between trajectories, this section treats how to compare the values of two *curvature-trees* to obtain the final similarity measure l_f for two trajectories. It is based on a three-staged approach:

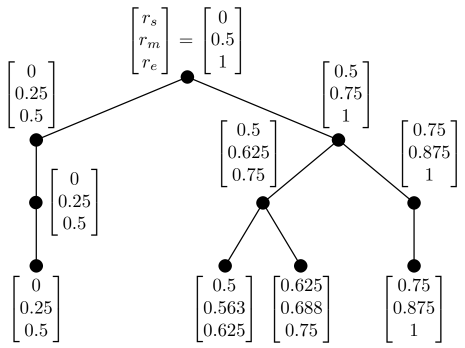
- Find corresponding nodes of the two *curvature-trees* to be compared.
- Normalize all scalar values stored in each node.
- Calculate the difference of each scalar value for every node and corresponding node across the two trajectories. Sum all differences up to obtain l_f.

The first point is a necessary condition for comparing *curvature-trees* with dissimilar structure. The second point allows one to calculate l_f as a sum of differences without the risk of one type of value (subspace projection length, PCA eigenvalue, condition number) having a too large or small influence. In addition, it is necessary to make the descriptor scaling invariant. Last, the third point gives us the similarity measure l_f.

For solving the correspondence problem, let the two trajectories to be compared be denoted *sample trajectory* (marked with a superscript ^{a}) and *reference trajectory* (marked with a superscript ^{b}). By introducing a mid-position $r_m \in [0, 1]$ representing the chronological order of all nodes of a certain *depth* value $d^{\{a,b\}}$, the position of each node $c^{\{a,b\}} \in \mathcal{C}^{\{a,b\}}$ is defined by the tupel $(r_m^{\{a,b\}}, d_{r_m^{\{a,b\}}})$. If the *curvature-trees* of both trajectories have a dissimilar structure, the corresponding node $c_o \in \mathcal{C}^b$, given $c_a \in \mathcal{C}^a$ is determined by

$$c_o = \underset{c^b \in \mathcal{C}^b}{\text{argmin}}(|r_m^a - r_m^b|) \quad \text{subject to} \quad d_{r_m^a} = d_{r_m^b}. \tag{8}$$

Determination of the value r_m for each node of the *curvature-tree* is performed by introducing two additional indices: The start- and end-position $\{r_s, r_e\} \in [0, 1]$. The algorithmic description with superscript ^{root} for the root node, ^{p} for every parent node, ^{c1} for every chronologically first child node and ^{c2} for every chronologically second child node is shown in Alg. 1.



$\begin{bmatrix} r_s \\ r_m \\ r_e \end{bmatrix} = \begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 0 \\ 0.25 \\ 0.5 \end{bmatrix}$
$\begin{bmatrix} 0.5 \\ 0.75 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 0 \\ 0.25 \\ 0.5 \end{bmatrix}$
$\begin{bmatrix} 0.5 \\ 0.625 \\ 0.75 \end{bmatrix}$
$\begin{bmatrix} 0.75 \\ 0.875 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 0 \\ 0.25 \\ 0.5 \end{bmatrix}$
$\begin{bmatrix} 0.5 \\ 0.563 \\ 0.625 \end{bmatrix}$
$\begin{bmatrix} 0.625 \\ 0.688 \\ 0.75 \end{bmatrix}$
$\begin{bmatrix} 0.75 \\ 0.875 \\ 1 \end{bmatrix}$

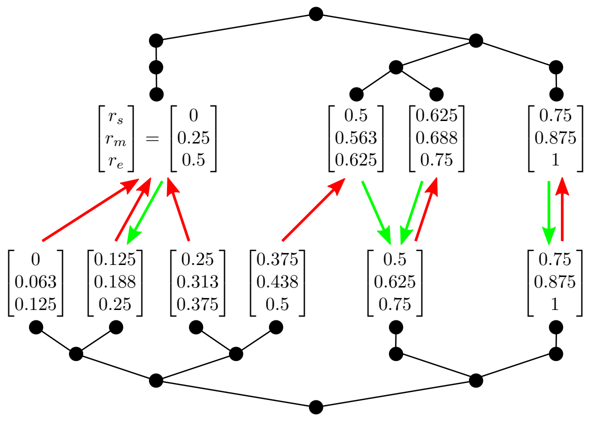
Fig. 5. Relative position for each node in the *curvature-tree*. Depending on the number of children nodes, the r_s, r_m and r_e values are either copied if there is only one children node or recalculated if there are two children nodes.

The goal of the normalization process is that subspace projection length, condition number and PCA eigenvalues

Algorithm 1: r_m allocation

```
Input: $\mathcal{C}$
Output: $\mathcal{C}$ with assigned $r_m$ values

foreach $c \in \mathcal{C}$ do
    if $c = c^{root}$ then
        $r^{root}_{\{s,m,e\}} = \{0, 0.5, 1\}$
foreach $c \in \mathcal{C}$ do
    if number of children = 1 then
        $r^{c}_{\{s,m,e\}}1 = r^{p}_{\{s,m,e\}}$
    else if number of children = 2 then
        $r^{c1}_{s} = r^{p}_{s}$
        $r^{c1}_{m} = (r^{p}_{s} + r^{p}_{m})/2$
        $r^{c1}_{e} = r^{p}_{m}$
        $r^{c2}_{s} = r^{p}_{m}$
        $r^{c2}_{m} = (r^{p}_{m} + r^{p}_{e})/2$
        $r^{c2}_{e} = r^{p}_{e}$
```



$\begin{bmatrix} r_s \\ r_m \\ r_e \end{bmatrix} = \begin{bmatrix} 0 \\ 0.25 \\ 0.5 \end{bmatrix}$
$\begin{bmatrix} 0.5 \\ 0.563 \\ 0.625 \end{bmatrix}$
$\begin{bmatrix} 0.625 \\ 0.688 \\ 0.75 \end{bmatrix}$
$\begin{bmatrix} 0.75 \\ 0.875 \\ 1 \end{bmatrix}$
$\begin{bmatrix} 0 \\ 0.063 \\ 0.125 \end{bmatrix}$
$\begin{bmatrix} 0.125 \\ 0.188 \\ 0.25 \end{bmatrix}$
$\begin{bmatrix} 0.25 \\ 0.313 \\ 0.375 \end{bmatrix}$
$\begin{bmatrix} 0.375 \\ 0.438 \\ 0.5 \end{bmatrix}$
$\begin{bmatrix} 0.5 \\ 0.625 \\ 0.75 \end{bmatrix}$
$\begin{bmatrix} 0.75 \\ 0.875 \\ 1 \end{bmatrix}$

Fig. 6. Correspondence of the *curvature-tree* nodes. Shown are corresponding nodes depending on their r_m-value for one *depth*-level $d = 4$. Red lines show the correspondence for the lower *curvature-tree* as the sample trajectory, green lines the correspondence for the upper *curvature-tree* as the sample trajectory.

are within comparable range of $[0, 1]$. For the subspace projection lengths, normalizing all $l_{\perp q}$ values with $2/l_s$ limits $l_s^{\{a,b\}}$ to be within $[0, 1]$ according to

$$l_s^{\{a,b\}} = \|\mathbf{t}_s - \mathbf{t}_{s1}\| + \|\mathbf{t}_{s2} - \mathbf{t}_s\|, \tag{9}$$

$$l_{\perp q}^{\{a',b'\}} = \frac{2l_{\perp q}^{\{a,b\}}}{l_s^{\{a,b\}}}, \quad q = 2, \dots, n. \tag{10}$$

Taking the inverse of the condition number κ normalizes it to $[0, 1]$ as shown in Eq. (11)

$$\kappa_q^{\{a',b'\}} = \frac{1}{\kappa_q^{\{a,b\}}}. \tag{11}$$

Concerning the PCA values, normalization is performed according to Eq. (12) by dividing all values with the largest eigenvalue

$$\lambda_f^{\{a',b'\}} = \frac{\lambda_f^{\{a,b\}}}{\max(\lambda_f^{\{a,b\}})}, \quad f = 1, \dots, n. \tag{12}$$

Calculating the difference is performed by comparing the normalized values for one node of the reference *curvature-tree* and corresponding sample *curvature-tree* as shown in Eq. (13) - (15). The denominator 2^d normalizes the cumulative influence of nodes of every *depth* value d.

$$l'_{d1} = \frac{\sum_q(|l_{\perp q}^{a'} - l_{\perp q}^{b'}|)}{2^d}, \tag{13}$$

$$l'_{d2} = \frac{\sum_q(|\kappa_q^{a'} - \kappa_q^{b'}|)}{2^d}, \tag{14}$$

$$l'_{d3} = \frac{\sum_f(|\lambda_f^{a'} - \lambda_f^{b'}|)}{2^d}. \tag{15}$$

By summing up all l'_{d1}, l'_{d2} and l'_{d3} values for every node of the *curvature-tree* and adding them, the final similarity measure l_f between two trajectories is obtained as shown in Eq. (16)

$$l_f = \sum_{\forall c^a \in \mathcal{C}^a} (l_{d1'} + l_{d2'} + l_{d3'}). \tag{16}$$

III. Experimental Evaluation

Experiments are performed in two ways: In the first set of experiments, the classification rate is evaluated using a proprietary dataset of recorded trajectories. In the second experiment the classification rate is compared with other methods using the Australian Sign Language dataset. For all experiments, it is w_L = number of sampling points of each (sub-)trajectory.

A. Classification of human free space motion

The first set of experiments is conducted using a RGB-D camera (Kinect) tracking a yellow marker being hold in one hand in 3D. The obtained dataset consists of 40 different movement classes as shown in Fig. 7 with 5 samples for each class. The 8 classes with a red boundary are conceptually similar to one of the other 32 "base" classes with the difference that the movement is paused at a certain point of the trajectory for a short time while still recording data. This way the temporal influence is investigated. All samples of a class are recorded varying the orientation, speed and scaling. Preprocessing of the data consists of an moving-average filter with a window size of 5 frames and a outlier removal for single points. The trajectories contain between 62 and 564 sampling points. The ROS openni-kinect interface is used for interfacing the camera. All other computations are conducted in Matlab R2010a using an Intel Core2Duo T7500 CPU with 4 GB RAM.

proc. d	parent	neighbor	time [s]	time 1:1 [us]
5	93.8%	90.6%	16.6	648
4	92.5%	91.3%	8.36	326
3	89.4%	89.4%	4.33	169
2	80.6%	80.6%	2.32	90.6
1	45.0%	53.1%	1.33	51.9

TABLE I
RESULTS EXPERIMENT 1: 3D TRAJECTORIES, 32 CLASSES, NO NOISE

For the first experiment only the 32 "base" classes (black sample trajectories in Fig. 7) are considered. Listed in Tab. I

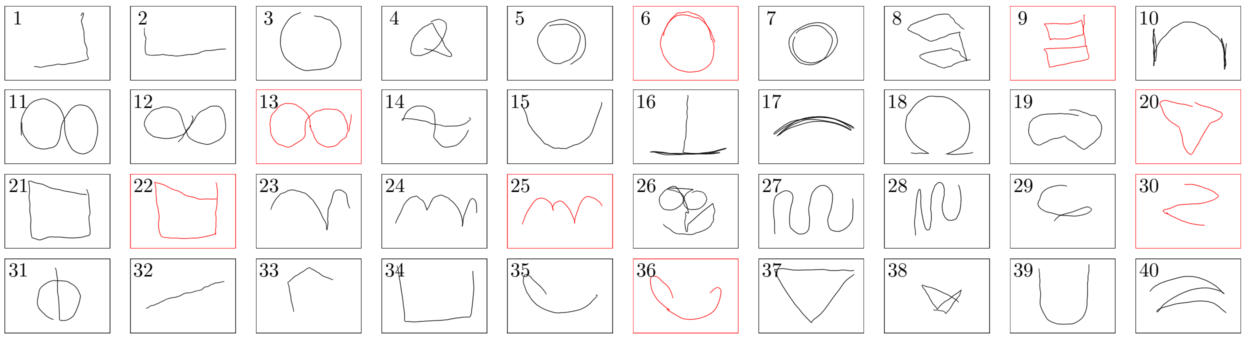


Fig. 7. Overview of the 40 different 3D classes used for classification. Each class consists of 5 samples. Every black trajectory belongs to one of the 32 “base’ ’classes. Each one of the other 8 red trajectory classes is a variation of one base class with paused movement at a specific point along the trajectory.

are from left to right: The maximum processed (compared) *depth* d of the *curvature-tree*, the percentage of correct classifications for the *parent method* and *neighbor method*, the average total time for comparing each trajectory with each other and the average time for comparing a single trajectory with another single trajectory. Here this equals the average total time divided by 160^2. It is observed that a good classification is already achieved for a maximum processed depth of 3 indicating most of the features are not encoded in the local but in the global properties of the trajectory.

proc. d	parent	neighbor	time [s]	time 1:1 [us]
5	86.5%	87.0%	25.8	645
4	86.0%	86.5%	13.5	338
3	81.5%	83.0%	7.00	175
2	73.5%	73.0%	3.58	89.5
1	37.5%	45.5%	2.04	51.0

TABLE II
RESULTS EXPERIMENT 2: 3D TRAJECTORIES, 40 CLASSES, NO NOISE

Classification results for the full dataset of 40 classes are shown in Tab. II. It displays a decrease in the correct classification rate compared to Tab. I of around 7%. Apparently, it is more difficult for the algorithm to distinguish between similar movements with varied recording speed than expected.

proc. d	parent	neighbor	time [s]	time 1:1 [us]
5	90.0%	90.6%	19.0	742
4	90.0%	90.0%	10.1	394
3	84.4%	87.5%	4.40	172
2	74.4%	77.5%	2.53	98.8
1	44.4%	48.8%	1.56	60.9

TABLE III
RESULTS EXPERIMENT 3: 6D TRAJECTORIES, 32 CLASSES, NO NOISE

The third experiment shows classification results for a set of 160 6D trajectories. In order to obtain 160 6D trajectories from the dataset of 160 3D trajectories, trajectories from two different classes are interpolated to have a similar number of sampling points and concatenated to form a 6D trajectory. Results are presented in Tab. III. The decrease in the correct classification rate is around 4% when being compared to Tab. I.

proc. d	parent	neighbor	time [s]	time 1:1 [us]
5	83.1%	80.6%	16.0	625
4	85.6%	85.6%	8.87	346
3	80.6%	80.6%	4.39	171
2	65.0%	68.8%	2.46	96.1
1	36.9%	40.6%	1.40	54.7

TABLE IV
RESULTS EXPERIMENT 4: 3D TRAJECTORIES, 32 CLASSES, WITH NOISE

Tab. IV displays results for the fourth experiment consisting of 160 trajectories in 3D with added Gaussian noise $\mathcal{N}(0, 0.002)$ for every dimension. Some examples of the noisy trajectories are shown in Fig. 8. Despite the extremely noisy trajectories, the correct classification rate drops only by around 10% compared to Tab. I. This sounds reasonable as it is explained earlier that for the given dataset the main differences of the trajectories are encoded in the global features and adding noise mostly affects local features.

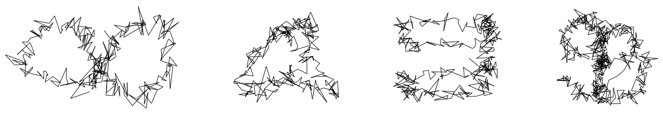


Fig. 8. Overview of some noisy trajectories for the fourth experiment.

B. Australian Sign Language Dataset

For a better comparison with other algorithms, our method is evaluated using the Australian Sign Language dataset (ASL)[1]. Similar to Croitoru [5], Vlachos [10] and (with reservation, as their classes are slightly different) Keogh [15], a set of 10 classes (“Norway”, “cold”, “crazy”, “eat”, “forget”, “happy”, “innocent”, “later”, “lose”, “spend”) with 5 samples per sign is processed. For every possible class pairing (45 in total), the 10 corresponding sequences are clustered through group-average agglomerative clustering. If the highest level of the resulting dendrogram separates

[1]`http://www.cse.unsw.edu.au/~waleed/tml/data/`

the two classes properly, clustering succeeded. The correct classification rates are presented in Tab. V.

proc. d	parent	neighbor	Croitoru	Vlachos	(Keogh)
5	62.2%	62.2%			
4	60.0%	57.8%			
3	51.1%	51.1%	53.1%	46.6%	51.1%
2	33.3%	37.8%			
1	2.2%	2.2%			

TABLE V
RESULTS EXPERIMENT 5: ASL DATASET

All displayed timings so far considered only the comparison of trajectories in case the *shape-tree* and *curvature-tree* are already created. As a reference: Creation of both trees takes on average around $400ms$ per trajectory for the proprietary dataset and around $60ms$ for the ASL dataset.

C. Discussion

Experiments show that the proposed approach has a high classification rate in various number of dimensions and is only slightly affected by noisy data. Considering computation time, the algorithm is efficient as creating the *curvature-tree* and comparing it can be performed independently. Consequently, the *curvature-tree* has to be created only once for each trajectory and can then be stored for further comparison. Another advantage is the lack of any tunable parameters such that the algorithm can be used out of the box. The real strength of the algorithm is based on the type of splitting process not just splitting each trajectory into two subtrajectories of equal length but finding an optimal *splitting point* through optimization of a shape matching problem.

One remaining problem considers distinguishing between similar movements with different temporal information where the classification rate drops noticeable. Another problem is the sensitivity to a proper trajectory segmentation. When creating the *shape-tree*, $\mathbf{t}_0$ and $\mathbf{t}_p$ form the fixed *start point* and *end point* of the topmost trajectory segment. Consequently, they have a huge influence on the overall classification result.

IV. Conclusion and Future Work

This paper presents a novel descriptor for trajectory classification in n-dimensional spaces which is invariant under scaling and rigid-body transformations. By iteratively splitting the trajectory up into smaller subtrajectories through optimization, a *shape-tree* stores the hierarchical description of every trajectory. Then the characteristic properties - subspace projection distance, condition number, PCA eigenvalues - of every trajectory and subtrajectory are measured at different granularity levels both at global and local scale and stored in a *curvature-tree*. This allows a quick comparison of different trajectories. The proposed algorithm succeeded well in classifying up to 40 different types of real-life trajectories in three and six dimensions with classification rates up to 94%. In addition, experiments performed on the ASL dataset show superiority when being compared to similar classification algorithms. Future work will be focused on an improved classification combining the proposed descriptor with more advanced features.

Acknowledgement

This work is supported in part within the DFG excellence research cluster *Cognition for Technical Systems - CoTeSys* (www.cotesys.org).

Appendix

Here we show that the proposed algorithm is invariant with respect to rigid transformations and scaling. As a rigid transformation does not alter the shape and size of the trajectory, i.e. preserves the distance between every pair of points, it does not affect the algorithm. To proof scaling invariance, let every point of the trajectory be multiplied by a scalar γ. Then Eq. (1) - (2) become

$$L_{0i} = \quad w_L \left(\sum_{k=0}^{i-1} \|\gamma \mathbf{t}_{k+1} - \gamma \mathbf{t}_k, \|\right)^2, \tag{17}$$

$$S_{0i} = \quad \sum_{k=0}^{i} \left\|\gamma \mathbf{t}_k - \gamma \mathbf{t}_k^{proj}\right\|^2. \tag{18}$$

Eq. (3) can be reformulated as

$$\mathbf{t}_s = \quad \gamma^2 \operatorname*{argmin}_{t_i \in \mathcal{T}_{0p}} (L_{0i} + S_{0i} + L_{ip} + S_{ip}), \tag{19}$$

and is therefore independent of the scaling factor γ. In addition, due to the normalization in Eq. (9) - (12) the final similarity measure l_f is independent of the scaling γ, too.

References

[1] A. Oikonomopoulos, I. Patras, M. Pantic, and N. Paragios, "Trajectory-based representation of human actions," in *IJCAI*, 2007, pp. 133–154.

[2] S. Calderara, R. Cucchiara, and A. Prati, "A dynamic programming technique for classifying trajectories," in *ICIAP*, 2007, pp. 137–142.

[3] J. R. Medina, M. Lawitzky, A. Mörtl, D. Lee, and S. Hirche, "An experience-driven robotic assistant acquiring human knowledge to improve haptic cooperation," in *IROS*, 2011, pp. 2416 –2422.

[4] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *KAIS*, vol. 7, pp. 358–386, 2005.

[5] A. Croitoru, P. Agouris, and A. Stefanidis, "3D trajectory matching by pose normalization," in *CIKM*, 2005, pp. 153–162.

[6] F. I. Bashir, A. A. Khokhar, and D. Schonfeld, "Segmented trajectory based indexing and retrieval of video data," in *ICIP*, 2003, pp. 623–626.

[7] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," in *CVPR*, 2009, pp. 312–319.

[8] J. De Schutter, "Invariant description of rigid body motion trajectories," in *ASME Journal of Mechanisms and Robotics*, vol. 2, 2010, pp. 1–9.

[9] S. Wu and Y. Li, "On signature invariants for effective motion trajectory recognition," *Int. J. Rob. Res.*, vol. 27, pp. 895–917, 2008.

[10] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *ICDE*, 2002, pp. 673–684.

[11] S. Wu, Y. Li, and J. Zhang, "A hierarchical motion trajectory signature descriptor," in *ICRA*, 2008, pp. 3070–3075.

[12] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering," *PVLDB*, vol. 1, pp. 1081–1094, 2008.

[13] J. gil Lee and J. Han, "Trajectory clustering: A partition-and-group framework," in *SIGMOD*, 2007, pp. 593–604.

[14] P. F. Felzenszwalb, "Hierarchical matching of deformable shapes," in *CVPR*, 2007, pp. 1–8.

[15] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for datamining applications," in *SIGKDD*, 2000, pp. 285–289.