Unsupervised Detection of Motion Primitives in very High Dimensional Sensor Data

Albert Hein and Thomas Kirste
Dept. of Computer Science
University of Rostock
18059 Rostock, Germany
{albert.hein, thomas.kirste}@uni-rostock.de

Abstract. The unsupervised identification of motion primitives in sensor data is widely seen as an important foundation of high-level activity recognition. Currently there are no clustering algorithms capable of processing massive, very high-dimensional data sets. In this paper we present an adapted projected stream clustering algorithm, which is able to efficiently detect motion primitives in subspaces of the feature space. The algorithm was tested on $> 7GB$ of motion capturing data from a home care scenario. We evaluated to what extent we were able to detect meaningful clusters using video annotation and real time rendering for visual analysis.

Keywords. Activity Recognition, Data Mining, Projected Clustering, Wearable Sensors, AAL

1. Introduction

Activity Recognition with wearable sensors has become a popular field of research during the last years due to its usefulness for applications, e.g. in the medical area, elderly care or ambient assisted living, and because of very promising first results, at least for simply structured motions. An important step which usually precedes the creation and training of classifiers or models is to look at the raw sensor data to find relevant and easily recognizable features and patterns where an algorithm can be tailored to. This is also essential for getting an understanding of the motion sequences, the sensor behaviour and accuracy, and for the analysis and interpretation of the final classification results. Here identifying postures or simple movements usually proves itself indispensable.

Looking further, the recognition of complex high-level activities also often assumes that these are compound of single postures or atomic actions. Unfortunately it is not known in advance, which ones are relevant for the disambiguation of certain activities, and, which is just as important, feasible to detect without unappropriate effort. This makes the manual selection difficult or even unmanageable.

If the data is getting too complex, those postures or actions are not visible to the naked eye anymore and data mining methods are needed for detecting them. A very special case is motion capturing, as this is an example of very high-dimensional sensor data

($>$ 500 features) at a very high rate ($>$ 100 data points / sec). Conventional clustering algorithms are not capable of handling such a high number of dimensions as here traditional similarity measures are not meaningful anymore. As we are not just interested in full body postures but perhaps certain arm gestures or head directions, different subsets of features are relevant for different postures. That makes global feature selection methods like Principal Component Analysis (PCA) unusable. Here usually projected clustering – again a relatively young field of research – can be utilized, where data points are clustered in variable subspaces of the data. Unfortunately, most algorithms are in their very early stages and are not able to gracefully handle this very large number of dimensions and can hardly ever process the massive amount of data arising while sampling at such a high rate as they get in trouble with performance, complexity and memory. The latter problem is addressed by stream data clustering algorithms, which only need a single run over the data as they are designed to work on continuously incoming data in an online manner. But to our knowledge none of these algorithms has been applied to motion sensor data, yet. For this reason we developed a very high performance projected stream clustering algorithm based on a modified version of HPStream [1].

This paper structures as follows: At first we will give an insight into related work concerning the unsupervised detection of postures or patterns in motion sensor data, as well as a short reflection of projected and stream clustering methods. Then we will explain our algorithm in detail. After that, the datasets used in this study will be presented and we will evaluate to what extend we were able to detect motion primitives using video annotation and real time rendering for visual analysis. To the end we will discuss advantages and drawbacks and give an outlook towards future developments and our next steps.

2. Related Work

The task of finding structures in motion sensor data without user intervention has been addressed in multiple approaches during the last years. In 2006 Huynh [2,3] suggested a method based on multiple eigenspaces for the unsupervised discovery of motion patterns in data from 4 acceleration sensors. He was able to detect clusters closely correlated to five base-level activities and combine this method with a Support Vector Machine for recognizing 8 different example activities in a semi supervised manner. He also presented an approach for discovering a small set of long-term activities of daily living using the concept of the Latent Dirichlet Allocation known from Topic Models [4]. His work has a slightly different focus than ours (at least in this paper), as he is generally interested in longer term patterns in the user's daily routine.

Nguyen et al. [4] used Hidden Markov Models (HMM) in combination with Gaussian Mixture models to find high level long term activities like sleeping, reading, office work, or driving based on the data of 3 acceleration sensors. He was able to recognize 12 activities and to detect unusual behaviour, both without supervision.

In our own approach [5] we combined a simple clustering algorithm (k-Means) with a Hidden Markov Model for detecting high-level care activities. We derived 562 features from 3 motion sensors, applied the pre-trained clusterer and used the cluster affiliations as observations for the generative model. The results showed the superiority of this unsupervised approach over classical discriminative classifiers (even when using the same model). Nevertheless, the clusters were defined globally using all dimensions and were impossible to interpret manually.

Clarkson et al. [6] used a hierarchy of HMMs to detect so called events (passing through doors etc.) and clustering them to scenes (like visiting the supermarket). Although their method works on audio and video data their modelling approaches are still interesting in the context of sensor data.

Finding patterns in time series data is often referred to as motif detection. As motion sensor data is a sort of multi dimensional time series data, these methods have also been applied in the past.

Hamid et al. [7] used n-grams for representing everyday activity classes and a graph based algorithm for finding recurrent activity patterns. They recorded over 5 months of indoor location data in a work and home scenario. However, their results are not intuitively understandable and difficult to interpret.

Minnen et al. [8,9] used one sensor for recording mock exercises and developed an iterative algorithm for finding motifs in that data. After initial results, they refined their algorithm and evaluated it on very short term actions in a 27 min data set. They later extended this approach to find motifs which only appear in subsets of the dimensions [10]. An approach where the elements of multi dimensional motifs may have temporal, length, and frequency variations has been proposed by Vahdatpour et al. [11] last year. They showed acceptable results both with synthetic and real world data.

Unsupervised learning for activity recognition has also been applied to other than motion sensors. Wyatt et al. [12] and Wang et al. [13], both from the same research group, use RFID object sightings for recognizing high level activities of daily living. HMMs and Dynamic Bayesian Models (DBN) are constructed and trained in an unsupervised fashion. They use common sense mined from the web for replacing explicit knowledge about the activity structures. These approaches are not suitable for finding short term patterns in motion sensor data.

All these approaches have some problems in common: Most of them are only able to detect coarse structures, and are working globally on all dimensions at the same time. Also most of them have been developed for low-dimensional data at a low datarate. As the majority of the algorithms is rather complex and mostly not linear with respect to dimensions, number of data points and number of clusters they are not applicable to our task.

Many dimensions and many data points are not just a performance problem: This is because the data in the high-dimensional case is very sparse, known as the curse of dimensionality. All data points tend to be almost equidistant from each other. In particular, traditional distance measures are not meaningful in the high-dimensional case. A recent technique is the *projected clustering* approach, which can determine clusters in a subset of the original dimensions. Each cluster has its specific particular group of dimensions, which reduces the sparsity problem to some extent. Even though a cluster may not be meaningful globally, subsets of dimensions can be found on which data points can form well-defined clusters. Generally said, projected clustering algorithms try to find the projection where the currently considered set of points clusters best. Each point is assigned to exactly one subspace cluster or is considered as noise. As said in the introduction, in our application different subsets of features are relevant for different postures. So the projected approach above seems more than appropriate. A good overview on clustering high-dimensional data is given in Kriegel et al. [14], where they discuss many projected, soft projected, subspace and hybrid clustering methods under various aspects.

The algorithm PROCLUS by Aggarwal et al. [15] despite beeing one of the first representatives of its class meets the requirements of our recognition task surprisingly well. The only problem seems to be that it needs multiple iterations over the data which may cause performance issues. Other projected, subspace and hybrid clustering methods described in [14] like PreDeCon (which is density based), CLIQUE and SUBCLU seem nearly optimally suitable, but their complexity grows with the dimensionality. Another major problem is, that these algorithms require the whole data set for random access in the memory. For many of those algorithms free implementations are available, so we tested them on our dataset[1]. We tried Open Subspace [16] in combination with Weka [17] and the ELKI framework [18] on a double Quad-Core Intel Xeon @ 2.26 GHz with 12 GB of RAM. Unfortunately we found that most algorithms either need a computation time of multiple days or simply abort with memory errors – or both.

The scope of *stream clustering* is finding structures in streaming data. Key requirements in this field, besides the classical criteria, are therefor linear computational complexity with growing number of incoming data points, memory efficiency and a high overall performance. Stream clustering is a very young field of research. A first survey from 2009 [19] covers only 5 algorithms. At the time of writing the only approach capable of clustering high-dimensional data is HPStream [1] by Aggarwal et al. which is a projected clustering algorithm. It combines the key ideas of PROCLUS with the requirements of clustering stream data. HPStream only needs one pass over the data, has a very memory efficient way for representing clusters and is able to adjust itself to long term changes in time in the data stream.

3. The Projected Stream Clustering Algorithm

The clustering method described in this section is a modified version of the HPStream algorithm [1] introduced by Aggarwal et al. in 2004. HPStream stands for High-dimensional Projected Stream clustering method and is distantly related to k-Means like approaches. Projected clustering lies within the paradigm of subspace clustering where each cluster has its own distinctive subset of dimensions which also may vary over time. HPStream is the first algorithm to introduce the concept of projected clustering to data streams and, at the time of writing, to the best of our knowledge the only stream clustering algorithm able to handle very high dimensional data gracefully. Additionally it pursues a linear update philosophy and only needs a single scan of each datapoint while still achieving a high clustering quality.

Key advantage of HPStream is its linear scalability in terms of dimensions, number of data points, and number of clusters. This is achieved by utilizing a very condensed cluster representation. Instead of saving every data point, it only keeps a few statistical attributes per cluster. These statistics are chosen in such a way, that updates can be done very efficiently even in fast data streams, while beeing still sufficient for quickly computing important measures about the cluster in each given projection. Further advantages of the algorithm are, that it can automatically determine the number of relevant clusters by itself and an implicitly integrated detection of outliers.

[1]For a detailed description see section 4

Another very important feature for stream data clustering that we won't use in this context is the algorithms ability to fade out the level of importance of old data and clusters over time and in this way to adapt to more recent trends. Therefor they introduced the Fading Cluster Structure using a special weight function. Here, we will introduce the cluster structure in a strongly modified way as we are working with static data, where fading out older clusters would lead to fatal misinterpretations.

Other modifications involve the way irrelevant clusters (and as we will see outliers) are beeing detected and removed as we had to adapt the algorithm to the activity recognition domain, the normalization preprocess (as we know all measurement ranges in advance), and the way the relevant number of projected dimensions per cluster is computed. Here, instead of picking a predefined number of dimensions with the least intra cluster variability, we are choosing a dynamic set using an ε threshold.

Cluster Structures

The data consists of a set of multidimensional data points $\overline{X}_1 \dots \overline{X}_k$. Each data point is a record containing d dimensions denoted by $\overline{X}_i = (x_i^1 \dots x_i^d)$. Now we will define the cluster structure used in our paper while trying to stay as close as possible to the original definition. Notice that we don't make use of a temporal fading function as the original paper does as explained before. The cluster structure aims at capturing a sufficient number of underlying statistics (first and second moments) so that it is possible to compute important characteristics of the particular clusters.

Definition 1. *A cluster structure for a set of d-dimensional points $C = \{X_{i_1} \dots X_{i_n}\}$ is defined as the $(2 \cdot d + 1)$ tuple $FC(C) = (\overline{FC2^x(C)}, \overline{FC1^x(C)}, W)$. The vectors $\overline{FC2^x(C)}$ and $\overline{FC1^x(C)}$ each contain d entries. Each of these sets of entries will be explained now:*

1. *For each dimension j, the jth entry of $\overline{FC2^x(C)}$ is given by the sum of the squares of the corresponding data values in that dimension. Thus, $\overline{FC2^x(C)}$ contains d values. The jth entry of $\overline{FC2^x(C)}$ is equal to $\sum_{k=1}^{n} (x_{i_k}^j)^2$.*
2. *For each dimension j, the jth entry of $\overline{FC1^x(C)}$ is given by the sum of the corresponding data values in that dimension. Thus, $\overline{FC2^x(C)}$ contains d values. The jth entry of $\overline{FC2^x(C)}$ is equal to $\sum_{k=1}^{n} x_{i_k}^j$.*
3. *We also maintain a single entry W containing the number of data points[2]. Thus, this entry is equal to $W = n$.*

An important property can be derived from the clustering structure described in Definition 1, which is referred to as *additivity*. It is defined as follows:

Observation 1. *Let C_1 and C_2 be two clusters with cluster structures $FC(C_1)$ and $FC(C_2)$ respectively. Then, the cluster structure of $C_1 \cup C_2$ is given by $FC(C_1 \cup C_2) = FC(C_1) + FC(C_2)$.*

This additivity property follows from the fact that each cluster can be expressed as a sum of its individual components and facilitates adding new data points (= clusters with one element) to an existing cluster.

[2]We chose to keep W as the notation for this number for compatibility reasons, although W does not describe a weighting function anymore as in the original algorithm.

As we aim at doing projected clustering, we associate an individual set of dimensions with each cluster. Like the original paper, with each cluster C, we associate a d-dimensional bit vector $B(C)$, which corresponds to the relevant dimensions in C. Each element in this vector has the value 1 or 0, whether the corresponding dimension in this cluster is included or not. This vector is needed for the assignment of new data points to the appropriate cluster. As the algorithm progresses, this vector changes, reflecting the varying set of dimensions.

Algorithmic Details

Next we want to introduce the clustering algorithm built upon this structure with its corresponding procedures. The algorithm itself is an iterative process, continuously adding incoming data points to existing cluster structures, adding new structures, and re-defining the set of relevant features in each cluster. Again, I will try to stay as close as possible to the original explanation.

Before this iterative process can start, a normalization step is needed to weight different dimensions correctly. As the algorithm is choosing the relevant dimensions of each cluster by comparing the radii (standard deviations along each dimension) along different dimensions, normalization is crucial, as all dimensions represent different measurements with different ranges and variances. We know all measurement ranges in advance, so we could modify this process to our needs. For each $\overline{X}_i = (x_i^1 \dots x_i^d)$ and for every dimension d we can compute the normalized value $x_i^{d'}$, as we know the absolute measurement range of each feature.

Algorithm 1 illustrates the basic iterative clustering process which is executed once for every data point. The parameters of the algorithm are the incoming data point $\overline{X}$, the current cluster structures FCS, the corresponding set of d-dimensional bitvectors BS (keeping 1 bit for every dimension included in cluster C_i), and a dimensionality threshold constant ε and a boundary scaling factor τ which will be explained later.

Algorithm 1 Basic Clustering Algorithm

```
Require: Data Stream Point: $\overline{X}$, Cluster Structures: $FCS$, Dimensionality Vector Sets:
  $BS$, Dimensionality Threshold $\varepsilon$, Boundary Scaling Factor $\tau$
  {receive next data point $\overline{X}$ from stream}
  $BS \Leftarrow ComputeDimensions(FCS, \varepsilon, \overline{X})$
  for $r = 1$ to $|FCS|$ do
    $ds(r) \Leftarrow FindProjectedDistance(FC^x(C_r), B(C_r, \overline{X}))$
  end for
  $index \Leftarrow argmin_i\{ds(i)\}$
  $s \Leftarrow FindLimitingRadius(FC^x(C_{index}), B(C_{index})) \cdot \tau$
  if $ds(index) > s$ then
    add new cluster structure $C_{|FCS|+1}$ with a solitary data point to $FCS$
  else
    add $\overline{X}$ to $FC^x(C_{index})$
  end if
  remove clusters with zero dimensions from $FCS$
```

Algorithm 2 ComputeDimensions

```
Require: Cluster Structures: $FCS$, Dimensionality Threshold $\varepsilon$, Data Stream Point: $\overline{X}$
  create $|FCS|$ (tentative) cluster structures by adding $\overline{X}$ to each of the existing clusters
  compute the $|FCS| * d$ radii of each of the $|FCS|$ (tentative) clusters along each of the $d$ dimensions
  pick all dimensions with a radius $< \varepsilon$
  create a bitvector $B(C_r)$ for each cluster $C_r$ reflecting its projected dimensions
```

Algorithm 3 FindProjectedDistance

```
Require: Cluster Structure: $FC^x(C_r)$, Bitvector: $B(C_r)$, Data Stream Point: $\overline{X}$
  {This Procedure finds Manhattan Segmental Distance along the projected dimensions}
  for all dimensions with bit value of 1 in $B(C_r)$ do
    find the distance between $\overline{X}$ and the centroid of $C_r$
  end for
  return arithmetic average distance along the included dimensions
```

Algorithm 4 FindLimitingRadius

```
Require: Cluster Structure: $FC^x(C_{index})$, Bitvector: $B(C_{index})$
  {Find the radius $r'$ of the cluster only along the projected dimensions}
  $r_j^2 = \overline{FC2^x(C_{index})_j}/W - \overline{FC1^x(C_{index})_j}^2/W^2$
  $R = \sum_{j \in B(C_{index})} r_j^2$
  Let $d'$ be the number of bits in $B(C_{index})$ with value 1
  $R = \sqrt{R/d'}$
  return $R$
```

The clustering algorithm tries to assign each data point to the closest cluster structure at each step of the iterative process. Therefor it utilizes a projected distance measure, which only includes the relevant dimensions of each cluster into the distance computation. Parallely, we have to continuously re-define the set of projected dimensions for each cluster in order to keep the radii of the clusters over the projected dimensions as low as possible. Because of this, the clustering process has to maintain not only the clusters FCS, but also the set of dimensions associated with each cluster BS. Now we will go through the pseudo-code in Alg. 1 step by step.

- The set of dimensions assiciated with each cluster is updated using the procedure *ComputeDimensions* (Alg. 2) which determines the dimensions in such a way that the spread along the chosen dimensions is as small as possible. As clusters may only contain few or one point the computation of the cluster's radii is unstable or even impossible. Thus, the incoming data point $\overline{X}$ is temporally added to each possible cluster during the determination of dimensions[3]. *ComputeDimensions* determins the $|FCS| * d$ radii (standard deviations) along all dimensions of each cluster in FCS. Now all dimensions with radii $< \varepsilon$ are selected. This allows the number of projected dimensions to vary between clusters and during the progress

[3]For a detailed explanation see [1]

of the algorithm. The intuition behind this is, that clusters represent areas in the subdimensional space with very low variability compared to the noise in other dimensions. After choosing the dimensions for each cluster the values are stored in the set of bitvectors BS.

- Next, the closest cluster to the incoming data point $\overline{X}$ is determined by computing the pairwise projected distances using the procedure *FindProjectedDistance* (Alg. 3) and choosing the minimum value. *FindProjectedDistance* calculates the Manhattan Segmental Distance between $\overline{X}$ and the centroid of the cluster C_r along each dimension with bit value 1 in $B(C_r)$. It is not necessary to normalize the distance measurements in this point, since the entire data set has been normalized in the preprocess.
- As we now selected the cluster C_{index} which $\overline{X}$ should be assigned to, we calculate its limiting radius as the natural boundary of the cluster. This is done by the procedure *FindLimitingRadius* (Alg. 4). This radius can be computed using the statistics in the cluster structure. As we are storing the first and second order moments of all included datapoints in each cluster structure, for every dimension j we can calculate the average square radius r_j^2:

  $$r_j^2 = \overline{FC2^x(C_{index})_j}/W(t) - \overline{FC1^x(C_{index})_j}^2/W(t)^2$$

  The square root of the sum of the radii along the projected dimensions given by $B(C_{index})$ is the limiting radius R and is calculated as $R = \sum_{j \in B(C_{index})} r_j^2$. This value is scaled by a boundary factor τ in order to decide the final value of the limiting radius of the cluster. Data points lying outside this boundary are not added to the cluster. Instead these points create new clusters of their own.
- If a new cluster is created, the total number of clusters in FCS increases. If the new data point $\overline{X}$ is an outlier, the newly created cluster will subsequently have few points added to it and be deleted ultimately.
- Whenever a data point is added to the statistics of the corresponding cluster, the additivity property ensures that the updated cluster is represented by these statistics.

As you may have noticed, it is necessary to perform an additional initialization step, since it is not possible to start the iterative process with an empty set of cluster structures FCS. In this step the original clusters are created using a certain initial number of data points with a conventional clustering algorithm. So first, a full dimensional k-Means algorithm is applied to the data points to create inital clusters. Then the procedure *ComputeDimensions* is applied in order to determine the most relevant dimensions for each cluster. After that, we start assigning each data point to the closest centroid according to the selected dimensions. Now a new set of k centroids is calculated and the process is repeated until it converges to a final number of clusters. These are used as the initial set of clusters for the projected stream clustering algorithm.

4. Dataset

Our algorithm was evaluated on motion capturing data recorded for the MARIKA project. Aim of this project is to support nurses in the elderly care with a mobile assistance system, among others by the online documentation of their care activities. This

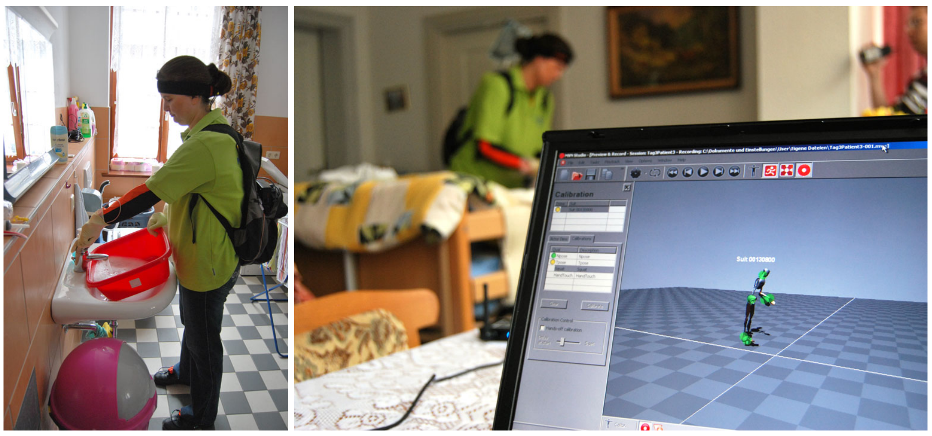


Figure 1. Left: The Nurse equipped with the full body motion capturing suit and RFID-readers on both hands. Right: The motion capturing software showing the rendered equivalent of the nurse during recording. An assistant is filming every activity on video for later manual annotation.

closed world scenario can be seen as an exercise example for detecting arbitrary activities of daily living. We equipped two professional nurses with a full body motion capturing suit, which, because of its inertial sensor technology, could be operated wirelessly in any environment and be worn under the normal clothing (see Fig. 1). We recorded three days of their usual care tour (one nurse at a time), where they did house calls at three different patients' homes. They were also wearing RFID-readers on both of their wrists, but we won't pay attention on these in this paper. Besides the nurse, two helpers were monitoring the recording and surveilling each activity by video for later annotation.

As sensor hardware we used an Xsens MVN2 inertial full body motion capturing suit[4]. As mentioned above, this suit is unique: Because of its motion tracking method using inertial measurement units, it does not depend on a camera infrastructure, can operate wirelessly and under clothes and is able to deliver highly sophisticated sensor data in realtime at a very high rate. It is equipped with 17 MTx 6DOF orientation trackers containing a 3d accelerometer, 3d gyroscopes, and 3d magnetometer each. The data was transmitted to a notebook pc carried in a backpack via 3 bluetooth channels, synchronized with the RFID sightings, and logged to disc.

We were able to collect 5 full house calls with a total length of 2 hours and 1 minute which makes 7.1 GB (!) of raw sensor data. Each dataset was recorded at a rate of 120 Hz with 793 features for each data point. These features consist of acceleration, angular velocity, magnetic field and orientation for each of the 17 sensors; orientation, position, velocity, acceleration, angular velocity and angular acceleration for 23 body segments; two angles for 22 joints, and the position of the center of mass. All these values either consist of 3 dimensional double vectors or quaternions. Low-level features are either directly measured or computed on the sensor boards with extended kalman filters, the derived features are calculated by the MVN biomechanical model.

Video footage and sensor data have been synchronized afterwards and manually annotated using ELAN[5], an open source tool for video annotation developed by the Max

[4]http://www.xsens.com/en/general/mvn
[5]http://www.lat-mpi.eu/tools/elan/

Planck Institute for Psycholinguistics in Nijmegen, the Netherlands. In ELAN it is possible to maintain multiple annotation tracks. We use one track for the manual labelling of the high-level care activities, two tracks for the RFID object sightings at both hands (automatically generated from our recording), and one track for the cluster memberships computed by our projected stream clustering algorithm. All care activities were taken from the service accounting catalogue of the health insurances. As we are only interested in the unsupervised detection of motion primitives in this paper, we will ignore the annotations hereafter.

5. Detection of Motion Primitives

We evaluated the projected stream clustering algorithm using the whole care dataset with full dimensionality. We only removed the absolute position data (mass point x, y, and z), as we did not want to learn locations, which would impair the robustness of the trained clusterer.

As described in the section above, the data set consists of 5 single large recordings. The clusterer was trained and evaluated via leave one out: We trained the algorithm on 4 of the full data sets and applied it on the remaining one. This procedure was repeated 4 times with rotating data sets. The leave one out strategy ensures realistic clustering results regarding the accuracy and robustness against unseen data. When the clusterer was applied, it assigned every single data point to the closest cluster (with respect to the clusters' dimensionalities). After preliminary experiments, the algorithm was parameterized as follows: We used $n = 2000$ data points for the initialization process as recommended by the original paper. The number of seed clusters for the init procedure was set to $k = 35$ following our own findings from [5]. We defined a minimal cluster size of 120 datapoints $\widehat{=} 1s$. That means each cluster must overall represent at least $1s$ (not obligatory connected) of the training data set. The parameters $\varepsilon = 0.001$ and $\tau = 3$ were determined experimentally.

For all five test runs, Table 1 shows the number of data points, the number of detected clusters, the average number of data points per cluster during training, the average number of dimensions per cluster, and the average duration of the assignments per cluster. The last column shows the corresponding means over all test runs.

data set	1	2	3	4	5	avg
data points	148326	201499	115532	122386	286704	
clusters	54	60	36	100	99	69.8
avg data points per cluster	274	250	333	251	252	272
avg dimension per cluster	540	508	501	521	518	517.6
avg duration per cluster	2.34s	1.75s	3.17s	1.74s	1.78s	2.16s

Table 1. Number of data points, the number of detected clusters, the average number of data points per cluster during training, the average number of dimensions per cluster, and the average duration of the assignments per cluster for all five test runs. The last column shows the corresponding means.

For the detailed evaluation of the clustering process and results we picked a small example data set taken from the original recordings (2 minutes from set 1). Again, we trained the clusterer on the data set, saved the cluster affiliations, and exported cluster statistics.

To get a feeling for the raw data, a small random bit from the example data set is shown in Fig. 2. The vertical axis represents 100 dimensions, the horizontal axis 240 data points, which cornesponds to $2s$. All values have been normalized and visualized as a greyscale bitmap. The data is vertically grouped by segments, sensors and joints. In the upper right corner we can see static areas with nearly constant values, some patterns look random, others cyclic. There is some noticeable action in the upper left corner of the image. The data set was recorded while the nurse was putting on her rubber gloves. The Original image for the example data set has a size of 793×14400 pixels and would appear as grey rectangle in this print. As one can imagine, its is nearly impossible to interpret even such a short timeframe of the data, the visual analysis is very inconclusive.

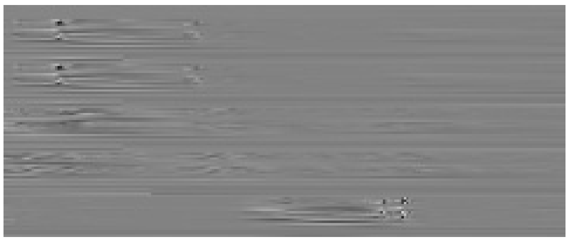


Figure 2. Excerpt from the raw data visualized as a greyscale bitmap. This figure shows 2 seconds of the example data set (horizontal) along 100 dimensions (vertical).

The clustering algorithm was trained and applied the same way as above, using the same parameters, except the training and test sets were the same. This is reasonable in this case as we are not interested in robustness but in the algorithm's behaviour and results in detail. The clusterer found 21 different clusters representing motion primitives. In average they were trained using 203.8 data points, contain 539 dimensions, and the cluster assignments have an average duration of $1.3s$. The label assignments over time are plotted in Fig. 3. The vertical axis shows the cluster label (the labels of the clusters are not ordered, they only represent when the specific cluster was created during training). A data point is always assigned to the closest cluster (with respect to the dimensions the cluster contains). There are very few different, very constant clusters on the left, in the middle there are rapidly changing clusters followed by a constant increase. Then again a rapid fluctuation appears, followed by continuously evolving clusters. It is not possible to get any further insights from this view except making higly speculative hypotheses.

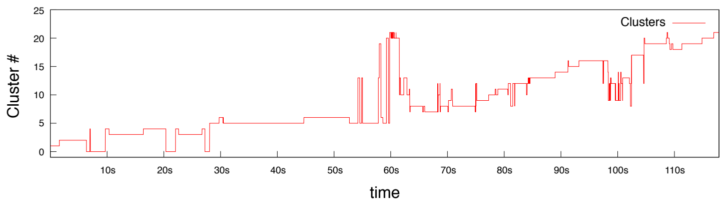


Figure 3. Cluster label assignments of the example data set over time.

Detailed statistics for each cluster are shown in Fig. 4. The plot shows the number of data points used during training, the dimensionality, and the average duration of cluster assingments. Especially clusters 3 and 4 seem to be very popular, while other clusters partially have very short assignments. If we compare this with Fig. 3, we see that these big accumulations correspond to the stable clusters in the beginning, the others occur in the middle and the end of the plot. Some assignments $< 1s$ are very short but often repeated. These seem to represent seldom but significant events.

The dimensionality always lies above > 300. This implied that most clusters are representing multiple (not mandantory connected) body segments doing a special motion. The first 3 clusters are full dimensional (this originates from the k-means initialization step starting with all dimensions) representing crisp full body postures. When configuring the algorithm, the overall dimensionality can be reduced by decreasing parameter ε. Again, further interpretation is difficult and mostly speculative.

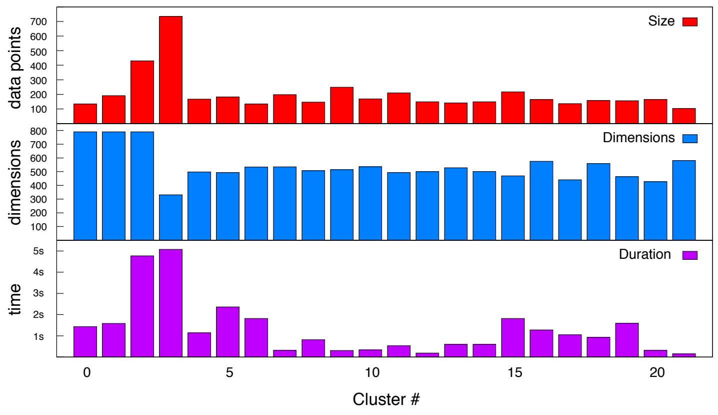
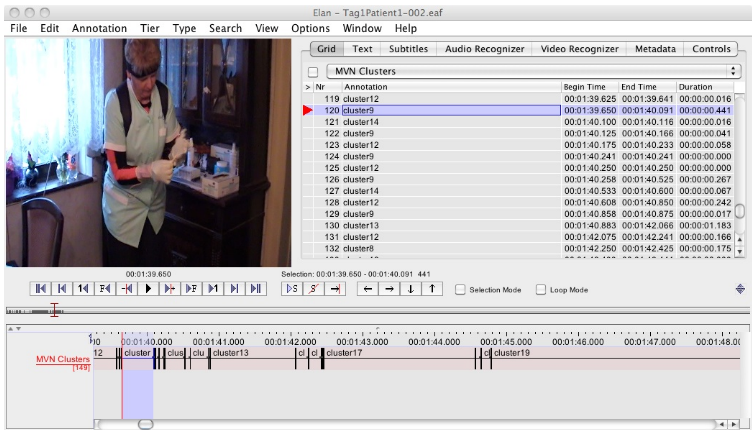


Figure 4. Cluster properties of the example data set. From top to down: Cluster size in number of overall assigned data points, number of dimensions and average duration of cluster assignments.

As the cluster names are not meaningful, the plots require much imagination and guessing. It is helpful to bring clusters in relation to the real story line. Hence we underlayed the clusters with our video footage and synchronized the time offsets. Therefore we exported the cluster label assignments over time to the ELAN annotation file format as a track of annotation using the cluster labels as single annotations. A screenshot is shown in Fig. 5. The video can be played in real time, slow motion or frame by frame and the clusters are following on a timeline, facilitating the navigation. The current position is indicated by a cursor. Cluster labels can also be viewed as subtitles. These features extremely simplify the interpretation of the motion primitives.

Looking back at Fig. 3 the plot becomes much clearer. In the beginning the nurse is standing nearly still listening to instructions (very few, very constant clusters on the left), in the middle she takes the gloves from a box (full body motion at around $60s$, rapid changing of clusters) and puts the first one on her right hand (takes longer than usual, only fore arm/hand/head movement, clusters evolving continuously). Then she is taking the second glove (coarser motion/rapid changing of clusters again) and puts it on her left hand until the end and beyond (clusters evolving continuously).

Although navigating the video along cluster labels facilitates the interpretation, it does not show the cluster structure itself. A 793 dimensional cluster centroid is not in-


Elan – Tag1Patient1–002.eaf
File Edit Annotation Tier Type Search View Options Window Help
Grid Text Subtitles Audio Recognizer Video Recognizer Metadata Controls
MVN Clusters
00:01:39.650
Selection: 00:01:39.650 - 00:01:40.091 441
Selection Mode
Loop Mode
MVN Clusters

Nr	Annotation	Begin Time	End Time	Duration
119	cluster12	00:01:39.625	00:01:39.641	00:00:00.016
120	cluster9	00:01:39.650	00:01:40.091	00:00:00.441
121	cluster14	00:01:40.100	00:01:40.116	00:00:00.016
122	cluster9	00:01:40.125	00:01:40.166	00:00:00.041
123	cluster12	00:01:40.175	00:01:40.233	00:00:00.058
124	cluster9	00:01:40.241	00:01:40.241	00:00:00.000
125	cluster12	00:01:40.250	00:01:40.250	00:00:00.000
126	cluster9	00:01:40.258	00:01:40.525	00:00:00.267
127	cluster14	00:01:40.533	00:01:40.600	00:00:00.067
128	cluster12	00:01:40.608	00:01:40.850	00:00:00.242
129	cluster9	00:01:40.858	00:01:40.875	00:00:00.017
130	cluster13	00:01:40.883	00:01:42.066	00:00:01.183
131	cluster12	00:01:42.075	00:01:42.241	00:00:00.166
132	cluster8	00:01:42.250	00:01:42.425	00:00:00.175

Figure 5. Screenshots of the ELAN annotation software showing the example data set. Cluster labels are synchronized with video footage .

terpretable by hand. Generally speaking, one single centroid represents averaged motion capturing data which could still be used for visualization. We implemented a MVN file format exporter for the cluster data. For each data point the cluster membership is calculated as described above and the corresponding centroid is written into the motion capturing file instead of the original values. That file can be opened in the motion capturing software and viewed like real recorded data (Fig. 6).

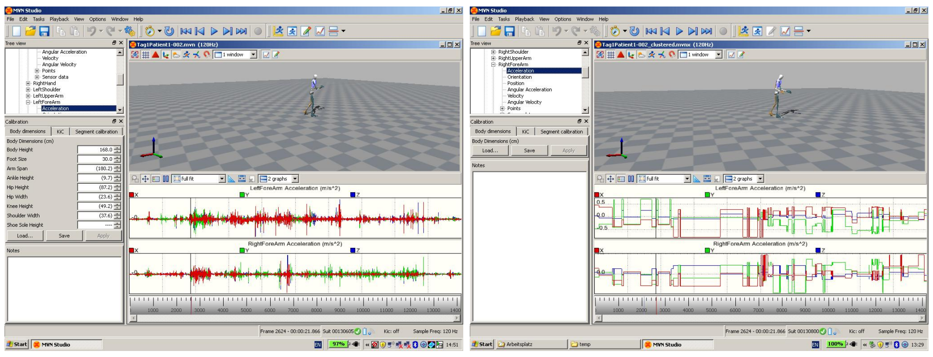

MVN Studio
LeftForeArm Acceleration (m/s^2)
RightForeArm Acceleration (m/s^2)

Figure 6. Screenshots of the motion capturing software showing the example data set at the same time stamp. Left: the originally recorded data. Right: data generated from the clusterer.

The left picture shows the originally recorded example data set at the same time stamp shown in ELAN (Fig. 5). The right side shows the same datapoint in the motion capturing file generated by the clusterer. Now it is possible to take a detailed look at single measurements: The plots in the screenshots show the acceleration of the fore arm segments. Notice the difference between the continuous values on the left and the right side, showing the same descrete segmentation from Fig. 3). Cluster centroids are visible as static postures in the MVN software. There is a slight difference between left and right picture although they show same frame. This is because the data in right image has been averaged over 169 data points during training.

When this fake recording is played in realtime, it looks like a coarse stop motion animation. Thus, the first impression is, that the clusterer detects static postures, but that appears to be wrong as most values do not represent static position, orientation, and angles but mostly acceleration, velocity, angular acceleration and angular velocity, which can not be visualized by the software. A surprising result is, that, already without temporal smoothing, the cluster affiliations are considerably stable over time and do not tend to flicker.

For this paper we did not evaluate the algorithm on artificial data using clustering quality measures. For a detailed discussion of the general behaviour and inherent limitations of the algorithm we refer to the original publication [1].

6. Conclusion

We modified an algorithm for clustering high-dimensional streams and adapted it for the unsupervised detection of motion primitives in very massive motion sensor data. The approach is following both, projected and stream clustering paradigms. As we found in our visual analysis, the algorithm seems to be able to find relevant clusters and dimensions autonomously. It is able to detect outliers and (if needed) able to operate online, so that old clusters can be adjusted to changes over time or get dropped, and novelties can be detected as new clusters. All these adaptions can be made with minimal changes to the algorithm. We evaluated the approach on real world motion capturing data taken from a high-level activity recognition experimental setting. It was able to identify stable motion primitives (postures and base level motion) without any user interaction. These can be used for the visual analysis of motion sensor data (in our example with video footage and 3d body animation). Furthermore, this method represents an intuitive way for dimensionality reduction, feature selection, and the selection of feasible sensor positions. For this task relevant dimensions or sensors can be simply selected according to the number of occurrences in the clusters. The unsupervised detection of motion primitives is widely seen as a major prerequisite for high level activity recognition. Probabilistic state-space models for instance, can adopt cluster affiliations as observation probabilities (we already demonstrated this in earlier work). Such a hybrid approach forms an effective generative method for detecting abstract activities. Our next steps would be to allow more complex shaped cluster representation, and not only hyperspheres as in the original algorithm. As seen in Fig. 2 besides static postures and short term motion primitives, the data contains temporal patterns of varying length, often referred to as motifs, which we also would like to detect with our approach. Simultaneously we are planning to combine the projected stream clustering algorithms with our models for high-level activity recognition.

Acknowledgements

The project MARIKA [20] is funded by the Ministry of Economics, Labour and Tourism of Mecklenburg-Western Pomerania, European Social Fund (ESF), European Regional Development Fund (ERDF).

References

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for projected clustering of high dimensional data streams," in *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pp. 852–863, VLDB Endowment, 2004.

[2] T. Huynh and B. Schiele, "Unsupervised discovery of structure in activity data using multiple eigenspaces," in *2nd International Workshop on Location- and Context-Awareness (LoCA)*, (Dublin, Ireland), Springer, May 2006.

[3] T. Huynh and B. Schiele, "Towards less supervision in activity recognition form wearable sensors," in *Proceedings of the 10th IEEE International Symposium on Wearable Computing (ISWC)*, (Montreux, Switzerland), October 2006.

[4] A. Nguyen, D. Moore, and I. McCowan, "Unsupervised clustering of free-living human activities using ambulatory accelerometry," *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, pp. 4895–4898, 22-26 Aug. 2007.

[5] A. Hein and T. Kirste, "Towards recognizing abstract activities: An unsupervised approach," in *Proceedings of the 2nd Workshop on Behaviour Monitoring and Interpretation, BMI'08, Kaiserlautern, Germany, September 23, 2008* (B. Gottfried and H. K. Aghajan, eds.), CEUR Workshop Proceedings, (Kaiserlautern, Germany), pp. 102–114, CEUR-WS.org, September 2008.

[6] B. Clarkson and A. Pentland, "Unsupervised clustering of ambulatory audio and video," in *ICASSP '99: Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference*, (Washington, DC, USA), pp. 3037–3040, IEEE Computer Society, 1999.

[7] R. Hamid, S. Maddi, A. Johnson, A. Bobick, I. Essa, and C. Isbell, "Unsupervised activity discovery and characterization from event-streams," in *Proceedings of the Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, (Arlington, Virginia), pp. 251–258, AUAI Press, 2005.

[8] D. Minnen, T. Starner, J. Ward, P. Lukowicz, and G. Troster, "Recognizing and discovering human actions from on-body sensor data," *Multimedia and Expo, IEEE International Conference on*, vol. 0, pp. 1545–1548, 2005.

[9] D. Minnen, T. Starner, M. Essa, and C. Isbell, "Discovering characteristic actions from on-body sensor data," in *Wearable Computers, 2006 10th IEEE International Symposium on*, pp. 11 –18, 11-14 2006.

[10] D. Minnen, C. Isbell, I. Essa, and T. Starner, "Detecting subdimensional motifs: An efficient algorithm for generalized multivariate pattern discovery," in *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, (Washington, DC, USA), pp. 601–606, IEEE Computer Society, 2007.

[11] A. Vahdatpour, N. Amini, and M. Sarrafzadeh, "Toward unsupervised activity discovery using multi-dimensional motif detection in time series," in *IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence*, (San Francisco, CA, USA), pp. 1261–1266, Morgan Kaufmann Publishers Inc., 2009.

[12] D. Wyatt, M. Philipose, and T. Choudhury, "Unsupervised activity recognition using automatically mined common sense," in *AAAI'05: Proceedings of the 20th national conference on Artificial intelligence*, pp. 21–27, AAAI Press, 2005.

[13] S. Wang, W. Pentney, A.-M. Popescu, T. Choudhury, and M. Philipose, "Common sense based joint training of human activity recognizers," in *IJCAI* (M. M. Veloso, ed.), pp. 2237–2242, 2007.

[14] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, pp. 1–58, 2009.

[15] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," *SIGMOD Rec.*, vol. 28, no. 2, pp. 61–72, 1999.

[16] E. Müller, S. Günnemann, I. Assent, and T. Seidl, "Evaluating clustering in subspace projections of high dimensional data," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 1270–1281, 2009.

[17] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems).* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[18] E. Achtert, H.-P. Kriegel, and A. Zimek, "Elki: A software system for evaluation of subspace clustering algorithms," in *SSDBM* (B. Ludäscher and N. Mamoulis, eds.), vol. 5069 of *Lecture Notes in Computer Science*, pp. 580–585, Springer, 2008.

[19] A. R. Mahdiraji, “Clustering data stream: A survey of algorithms,” *Int. J. Know.-Based Intell. Eng. Syst.*, vol. 13, no. 2, pp. 39–44, 2009.

[20] “Landesforschungsverbund mobile assistenzsysteme http://www.mobile-assistenzsysteme.de/,” April 2010.