MASARYK
UNIVERSITY

FACULTY OF INFORMATICS

Advancing Motion Words for Human Motion Classification

Master's Thesis

BC. DAVID PROCHÁZKA

Brno, Spring 2023

MASARYK UNIVERSITY

FACULTY OF INFORMATICS

Advancing Motion Words for Human Motion Classification

Master's Thesis

BC. DAVID PROCHÁZKA

Advisor: doc. RNDr. Vlastislav Dohnal, Ph.D.

Department of Machine Learning and Data Processing

Brno, Spring 2023

FACULTAS ARTIS INFORMATICÆ • UNIVERSITAS MASARYKIANA •

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Bc. David Procházka

Advisor: doc. RNDr. Vlastislav Dohnal, Ph.D.

Acknowledgements

This endeavor would not have been possible without my advisor, Vlastislav Dohnal. I am grateful to him for his endless stream of compelling ideas, witty remarks, and zeal for research. I would also like to thank my family for their unwavering support.

Abstract

The thesis improves the technique for representing short motions called motion words. We present a new quantization technique called composite motion word by dividing the skeleton into non-overlapping body parts. The problems of inefficient indexing of motion word sequences and action repetitions are addressed by employing edit distance and its adaptation. These advancements achieve a classification accuracy of 81.75% evaluated on the HDM05 dataset. Based on the outcome of the action classification task, we design a two-stage classification framework that supplements the global classifier with specialized classifiers. The specialization employs two strategies intended to mitigate misclassifications that occur at the global level. The first method learns body parts based on which we can better discriminate between categories. The second method finds the most significant moment in the motion and compares the actions based on the neighborhood around the extrema. Our solution achieves an accuracy of 90.02% evaluated on the modified HDM05 dataset. Finally, for the PKU-MMD dataset, we introduce an extension of the composite motion word called joint relation, which models the mutual interaction between a set of joints.

Keywords

motion capture, MoCap, human motion data, motion words, action classification, body parts

Contents

1 Introduction 1

2 Human Motion Data Processing 2

2.1 Motion Words . . . 3
2.1.1 Skeleton Normalization . . . 4
2.1.2 Skeleton Sequence Segmentation . . . 5
2.1.3 Similarity of Segments . . . 6
2.1.4 Quantization into Motion Words . . . 6
2.1.5 Application Evaluation . . . 8
2.1.6 Improvements . . . 10

3 Composite Motion Word 12

3.1 Definition . . . 12
3.2 Joint Relations . . . 14
3.3 Sequence Alignment of Motion Words . . . 14
3.4 Action Repetitions . . . 15
3.4.1 Extending Composite Motion Word with Length 16
3.4.2 Concatenation of Motion Word Sequences . . . 16
3.4.3 Discussion . . . 17

4 Classification Framework 18

4.1 Motivation . . . 18
4.2 Architecture . . . 19
4.3 Classification Process . . . 20
4.4 Representing Misclassifications . . . 21
4.5 Selected Body Parts . . . 24
4.6 Finding Extremum . . . 25
4.7 Extremum Neighborhood . . . 28
4.8 Tuning Specialized Classifier . . . 30
4.9 Commentary . . . 31

5 Evaluation 34

5.1 Datasets . . . 34
5.1.1 HDM05 . . . 34
5.1.2 PKU-MMD . . . 34

5.2 Composite Motion Word 35
5.2.1 Clustering Body Parts 35
5.2.2 Sequence Alignment of Composite Motion Words 37
5.2.3 Action Repetitions 39
5.3 Classification Framework 39
5.3.1 Analysis of Classification Performance 40
5.3.2 Performance Impact of Classification Methods . 42
5.3.3 Merging Similar Categories 44
5.3.4 Joint Relations 45

6 Conclusion 48

A Attachments 50

Bibliography 51

List of Tables

5.1 Application performance of MWs on the HDM05-130 dataset with respect to parameters k and m using DTW to compare the MW sequences. 37

5.2 Comparison of the application performance of the best-performing CMWs and Hard MW on the HDM05-130 dataset with DTW and edit distance. m is equal to two. . . 39

5.3 The achieved application performance of $d_{repeated}$ distance compared to d_{edit} on the best-performing configurations. Evaluated on the HDM05-130 dataset. 40

5.4 Classification accuracy of CMW and the two-stage classification framework with the listed modifications. Evaluated using 2-fold cross-validation. 43

5.5 Classification accuracy of 10-fold cross-validation on the HDM05-65 dataset. Composite Motion Word and Two-Stage Classification framework evaluated with parameters $k = 250, t = \infty$. 45

5.6 Evaluation comparison of CMW and two-stage classification with and without the three relations in the CS and CV settings of the PKU-MMD dataset. 46

List of Figures

2.1 A skeleton. Adapted from [7]. 3

2.2 Skeleton normalization. Adapted from [13]. 4

2.3 Illustration of sequence segmentation. The segment shift is illustrative. The shift size is typically about 20% [1] to 25% [14] of the segment length. 5

3.1 Division of a skeleton into body parts for CMW. Adapted with modifications from [7]. 13

4.1 An example of a specialized classifier created based on the result of the ExtractFromTransposedGraph algorithm, classifying actions a, b, and c. The classifier receives actions classified by the global classifier into the invocation category *grabHighR*. Each action is then classified into one of the classification categories in CC. 24

4.2 Matrix M used in the PCA computation in the SelectExtremumNeighborhood algorithm. The number of poses is denoted by m, the number of joint coordinates by n, P^j for $j \in \{1, 4, 7, \ldots, n-2\}$ represents xyz-coordinates of a single joint, and P_i for $i \in \{1, \ldots, m\}$ represents a pose. . . 28

5.1 Classification accuracy of CMW on the HDM05-130 dataset with respect to parameters k and m using DTW distance to compare MW sequences. 36

5.2 Classification accuracy of CMW on the HDM05-130 dataset with respect to parameters k and m using edit distance to compare MW sequences. 38

5.3 A part of the misclassification graph for categories *grabHighR* and *depositHighR* evaluated on the HDM05-130 dataset. 18 actions in *grabHighR* are classified as *depositHighR*. 16 actions in *depositHighR* are classified as *grabHighR*. The rest are misclassifications into other categories. 41

1 Introduction

Advances in pose estimation software from ordinary videos enable even faster adoption of human motion applications. These range from protecting cars in the parking lots by classifying human motion against a database of standard and suspicious behavior to detecting falls in nursing homes so that appropriate action can be taken. As the amount of data increases, efficient processing becomes an issue, as these high-dimensional data incur non-trivial storage and processing costs.

In [1], the authors introduced a compact feature called motion word (MW), representing a short motion. Three quantization techniques were presented, with the main effort put into addressing a border problem. A sequence of MWs can be used in various tasks like searching for similar motions in a database or assigning a motion to an existing category. The properties of the initially proposed distance restrict the indexability of such sequences.

This thesis aims to follow up on the issues and directions yet to be considered. We discuss and address the problematics of MW sequence indexing in conjunction with a proposal for a new quantization technique envisioning the MW as a collection of body parts. In this thesis, we propose a solution to the MW indexing problem. We introduce a new MW quantization technique with an extension called joint relations and a distance function addressing action repetitions. Our contribution to the classification task is a two-stage classification framework with two classification methods. The MW and the framework are evaluated on two datasets with an implementation written in Java using MESSIF [2].

The processing of motion data, its representation, and especially the definition, creation, and application evaluation of MWs are discussed in Chapter 2. This is followed by a proposal for a new MW and sections on indexability and action repetition. The next chapter describes a classification framework, the design, the algorithms, and a commentary. The evaluation of the introduced MW and the framework is the topic of Chapter 5. The last chapter then concludes this thesis with possible future research directions.

2 Human Motion Data Processing

The first part of this chapter, based on a survey paper [3], briefly describes motion-capturing techniques, the representation of motion, and its applications. Subsequently, a compact representation is described in detail. Improvements suggested in other works are also present.

Human motion data is a particular type of data called spatio-temporal, where a human body is captured throughout time and space. The motion is captured in discrete timestamps, which goes in hand with how specialized devices operate. Vicon[1] can capture movement in hundreds of frames per second, it uses tens of optical sensors, and it can somewhat resist occlusion [4]. Kinnect v2[2] captures standard 30 frames per second, uses RGB and infrared radiation depth sensors and provides no occlusion resistance. Estimating a human body position from a general video is also possible thanks to recent advances in pose estimation software [5, 6].

As for the spatial side of the motion representation, the human body is not captured as a whole. Instead, each device simplifies the body by restricting the tracking to predefined body joints. The selection and the joint count are specific to each device. A collection of 3D coordinates of the tracked joints is called a *pose*. In order to visualize a single pose, a *skeleton*, a stick-man-like figure, is used. It adds connections between joints, similar to bones as in the human counterpart; see Figure 2.1.

The devices produce a sequence of poses with a given sampling frequency. We, therefore, talk about *skeleton sequences*. Formally, a skeleton sequence $S = (P_1, P_2, \ldots, P_n)$ is a sequence of poses, where $P_i \in \mathbb{R}^{j \cdot 3}$ represents the 3D skeleton configuration estimated at the time instant $1 \leq i \leq n$ consisting of the xyz-coordinates of j tracked joints. The skeleton sequences may be finite or infinite, but for purposes of this thesis, we restrict ourselves exclusively to the finite case. Finally, a short semantically defined sequence is called an *action*, e.g., waving

1. `https://www.vicon.com`
2. `https://learn.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows`

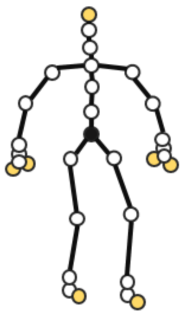


Figure 2.1: A skeleton. Adapted from [7].

with the right hand or sitting down. Visualization and exploration of motion sequences are discussed in [8].

When dealing with such data, its applications are particularly interesting. Each application task expects a certain type of input and imposes the type of the answer. *Action classification*, also called *action recognition* [9, 10], is dominated by machine learning techniques, especially neural networks, for their high accuracy. It focuses on assigning a previously unseen action to a semantically defined category. *Action detection* and the associated *action prediction* [11, 12] detect and possibly predict occurrences of a particular action in a sequence. *Search* task in a database of actions searches for the most similar ones based on a user query. Lastly, the *sub-sequence search* task takes a user query and searches for a similar sub-sequence motion occurrence in a database of sequences.

2.1 Motion Words

This thesis focuses on and extends the technique of motion words [1] (MWs). This section describes the steps necessary to convert a skeleton sequence into a sequence of MWs. We describe the original MW quantization techniques and note improvements made in other works.

The motivation for defining a MW stems from the need for more efficient processing of skeleton sequence data. The data is usually represented by high-dimensional vectors describing the 3D coordinates

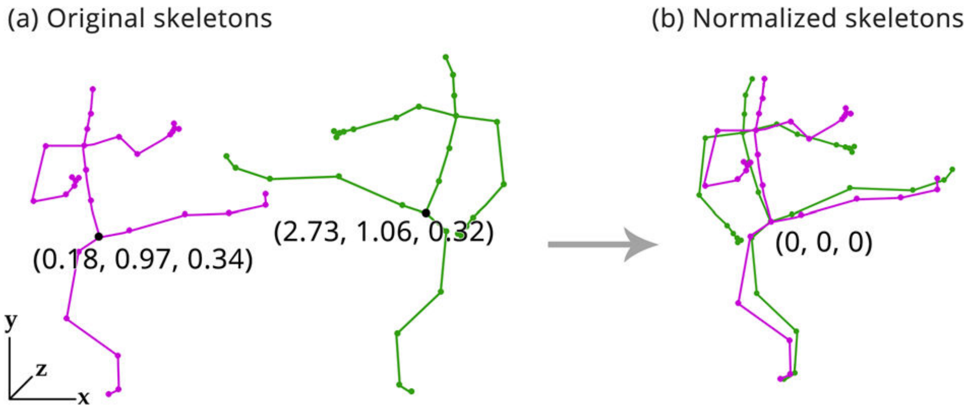


Figure 2.2: Skeleton normalization. Adapted from [13].

of each skeleton joint. It would be beneficial to represent this sequence more compactly.

The idea of MWs is to partition a skeleton sequence into fixed-size segments and replace each segment with an identifier of its canonical representation. This process produces a new representation, a sequence of one-dimensional identifiers we can think of as a text document. Each identifier, called a motion word, is conceptualized as a descriptor of a short motion sequence on which standard text retrieval techniques can be applied. Therefore, by using vector quantization, we can drastically reduce the storage requirements for each sequence and potentially improve performance in application tasks.

We describe the entire conversion process in detail in the following subsections. The skeleton sequence transformation into MWs is adapted from [1]. The process can be considered a multi-step pipeline invoked when a user enters a new sequence. The pipeline covers input data processing, normalization, frame rate reduction, segmentation, quantization, and translation to MWs.

2.1.1 Skeleton Normalization

The input skeleton sequence must be normalized to process a diverse range of people, mainly accounting for differences in height and length of legs and arms. We utilize a specific variant of the normalization tactics presented in [13]: *all-poses position normalization*, *all-poses ori-*

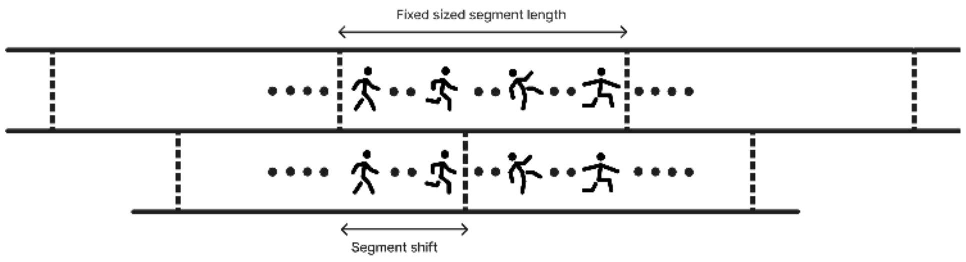


Figure 2.3: Illustration of sequence segmentation. The segment shift is illustrative. The shift size is typically about 20% [1] to 25% [14] of the segment length.

entation normalization, and *normalized skeleton*. We now describe the process in more detail.

We normalize the position, orientation, and size of the skeleton. The normalization of position is done by taking the root joint (the black joint in Figure 2.1) and shifting it together with the rest of the skeleton joints into position with coordinates $[0, 0, 0]$. The transformation is applied to all skeletons in all sequences. The orientation normalization is based on the left and right hip (the left and right joints under and connected to the root joint in Figure 2.1). Based on the position of the hips, an angle is computed for each skeleton. Every skeleton is then rotated by this angle such that during the whole motion, every skeleton faces the same fixed direction. Finally, we normalize the size of each skeleton based on average bone lengths across the whole population to obtain the same skeleton size for an adult as we would for a baby. The normalized skeleton is depicted in Figure 2.2. Note that each pose is normalized independently from the others.

2.1.2 Skeleton Sequence Segmentation

In this step, we cut the normalized skeleton sequences into overlapping fixed-sized segments. We specify two parameters: the number of poses per single segment and the number of overlapping poses between two consecutive segments. The first segment is taken from the start of the sequence. Every next segment is created by shifting the segment's start by the number of shift poses. A set of all segments forms a *segment* *space*. It contains all short-sized motions extracted from all motion sequences.

Some skeleton motions are captured with a high sampling frequency [3]. To also normalize the sampling frequency, we downsample it to values around ten frames per second. If the segmentation and downsampling parameters are set appropriately, we should see reduced computational cost without any significant loss of information [14]. Typically, a segment represents about 0.66 seconds of motion with a shift of 25% of the segment size.

2.1.3 Similarity of Segments

To assess the similarity between the two segments, we must consider the differences in performing the same motion. While normalization is primarily concerned with the size, orientation, and position of the skeleton, here we must focus on the execution of the entire motion. The motion can be performed at different speeds by different or the same person. To tackle the issue of temporal differences, the original paper used DTW algorithm [15], which we denote as d_{DTW}. It can exploit the speed variance between consecutive poses, but one must remember that the distance produced by DTW does not satisfy a triangle inequality[3]. Therefore, we cannot utilize numerous metric indexing structures.

The algorithm employs dynamic programming to determine the optimal sequence alignment based on the distance of the sequence elements. The internal distance, in our case, is computed based on a pair of poses. We quantify the distance by summing the Euclidean distances between the 3D coordinates of the corresponding joins.

2.1.4 Quantization into Motion Words

Near regions of the segment space represent similar motions. The goal of quantization is to exploit this property and partition the segment space into non-overlapping clusters. Each cluster contains similar

3. Proof by counterexample. Suppose three integer sequences $x = (0)$, $y = (1)$, $z = (1, 1)$ such that the distance between two integers is their absolute difference. Then $d_{DTW}(x, z) > d_{DTW}(x, y) + d_{DTW}(y, z)$ as $d_{DTW}(x, z) = 2$, $d_{DTW}(x, y) = 1$, and $d_{DTW}(y, z) = 0$.

segments that can all be treated as one representative segment. In other words, the cluster can be represented by a canonical segment. To minimize the spatial requirements for the representation of the canonical segment, we can assign a unique identifier to each such segment. It dramatically decreases the amount of data required for its representation. Note that we have to store the canonical segments to quantize new data.

Thus, the quantization mainly comprises the pivot selection by a clustering algorithm and the subsequent assignment of pivot identifiers to segments. For this purpose, we utilize the k-medoids algorithm [16]. It differs from the k-means algorithm [17] by producing pivots that are elements of the clustered space. Thus, we can define an MW vocabulary as a set of pivot identifiers selected by the clustering algorithm, called *motion words*, together with a Boolean-valued MW matching function $match^{MW} : MW \times MW \rightarrow \{0,1\}$. When two MWs match, the function returns one, otherwise zero.

Three techniques for translating skeleton sequences to MW sequences were originally introduced in [1]: *hard*, *soft*, and *multi-overlay*. Each segment is transformed into the corresponding MW representation and then placed back into the original skeleton sequences, replacing the segment.

Hard quantization produces a single identifier for each segment but does not address a border problem. The *border problem* is a situation where two segments are close but fall into different clusters, thus making them different in the eyes of the MW matching function.

Soft quantization tackles this issue by extending a single identifier into a vector of identifiers. It contains the first base identifier and extended identifiers. The extended identifiers correspond to identifiers of close clusters to the base cluster. We say the two soft MWs match if there is at least one base element in the intersection of their vectors.

Multi-overlay quantization handles the border problem by performing the clustering multiple times. Two MWs then match if the corresponding identifiers match in at least m cases of the n clusterings.

Finally, to quantify the distance between two MW sequences, DTW is used. Within DTW, the matching function decides the distance between two MWs. The distance function returns zero in case of a match and a distance of one in case of a mismatch. For two MWs

$A, B \in MW$, we formalize the distance between them as:

$$d_{MW}(A, B) = \begin{cases} 0 & \text{if } match^{MW}(A, B) = 1 \\ 1 & \text{otherwise} \end{cases}$$

2.1.5 Application Evaluation

The following process culminates into two different applications. The first is the action search, where the user searches for similar actions. The second application is action classification, where the user receives information about which of the existing action categories the query action belongs to. In this thesis, we focus mainly on the second application.

Search

The search is evaluated using kNN queries [18, p. 16]. The k is dynamically adjusted for each action based on the number of actions in the same category as the query action. The k is reduced by one because the query action is not considered, i.e., the leave-one-out approach. *Recall* is a ratio of the number of actions from the same category in the answer divided by the number of retrieved actions. The search *effectiveness* is then measured as an average recall over all actions.

Classification

In the classification task, the goal is to classify a new, previously unseen action into one of the categories. The straightforward yet effective approach utilizes the 1NN classifier [19, 13]. It searches for the nearest neighbor, excluding the query action, if present in the database, and returns the neighbor's category as the answer.

In this thesis, we employ the *Weighted-Distance kNN Classifier* as presented in [20], the same classifier used in [1]. It is a general probabilistic classifier formally defined as

$$classify : \mathcal{S} \times \mathcal{T} \times \mathbb{N} \rightarrow \mathcal{C},$$

which assigns to a general sequence $q \in \mathcal{S}$ a category $C \in \mathcal{C}$. The sequence q may belong to either a set of skeleton sequences or a set of MW sequences, both of which we denote $\mathcal{S}$ as the input domain of the sequences. $\mathcal{T}$ denotes the input domain of the training sequences where each sequence is assigned to some category in $\mathcal{C}$. The third argument $k \in \mathbb{N}$ corresponds to the number of neighbors taken into account during the classification decision.

The Weighted-Distance *k*NN Classifier [20] is designed to deal with the drawbacks of a 1NN classifier. Specifically, it aims to tackle the situation where multiple objects are at almost the same distance from the query object, each belonging to a different category. It becomes more of a prominent phenomenon in the case of MW sequences, as their distance is integral as opposed to the skeleton sequences, which produce real-valued distances. It works by considering a combination of a majority vote of the nearest neighbors and their similarity to the query object. First, the *k*NN query is defined to obtain k nearest neighbors to a query skeleton sequence or MW sequence q from the training set of sequences $T \in \mathcal{T}$:

$$\begin{aligned} kNN(q, T, k) = \{n \in N \mid\ & N \subset T, |N| = k, \\ & \forall n \in N, \forall s \in T \setminus N : \\ & d(q, n) \leq d(q, s)\} \end{aligned}$$

The *k*NN function is extended by an additional category parameter C, which filters the results by keeping only those neighbors belonging to the category C:

$$kNN(q, T, k, C) = \{n \in kNN(q, T, k) \mid n \in C\}$$

The relevance of each neighbor is computed by the function $compRel : \mathcal{T} \rightarrow [0, 1]$:

$$compRel(s) = 1 - \frac{d(q, s)}{dist \cdot weight},$$

$$dist = \max\{d(q, s) \mid s \in kNN(q, T, k)\},$$

where *dist* is the distance to the *k*th neighbor and *weight* is set to 1.1. The weight puts considerable significance on the distances while decreasing the relevance of the *k*-th neighbor [20]. The relevance of each category is then computed as:

$$compRels(q, T, k) = \{(C_i, r_i) \mid C_i \in \mathcal{C},$$
$$r_i = \sum_{n \in kNN(q,T,k,C_i)} comRel(n)\}$$

The relevance is normalized to obtain a probability and further ascertain they sum to one:

$$normRels(\{(C_1, r_1), \ldots, (C_m, r_m)\}) =$$
$$\{(C_i, p_i) \mid i \in [1, m], p_i = \frac{r_i}{\sum_{j=1}^{m} r_j}\}$$

Finally, the Weighted-Distance *k*NN classifier is defined as:

$$classify(q, T, k) = C_{max},$$

$$(C_{max}, p_{max}) = \operatorname*{argmax}_{(C_i,p_i) \in normRels(compRels(q,T,k))} p_i$$

The normalization result is a set of categories and their probabilities, indicating the probability that the query should be assigned to the corresponding category. While in the original paper [20], the result is defined as the above set, in this thesis, we consider the result to be only the category from the tuple with the highest probability.

In the following chapters, we distinguish between two classification methods, which differ in their internal decision process. To simplify the notation of the *classify* function, we use the following functions instead: $\mathcal{G} : \mathcal{S} \to \mathcal{C}$ and $\mathcal{S}_{SC} : \mathcal{S} \to \mathcal{C}$. $\mathcal{S}$ and $\mathcal{C}$ correspond to the same parameters as in the *classify* function. The values of T and k are set during the evaluation.

2.1.6 Improvements

The authors of [21] studied a different approach to the quantization phase. They employed complex density-based algorithms such as DBSCAN [22], HDBSCAN [23], or Jarvis-Patrick [24], but could not improve the application performance. While the evaluated algorithms produced high-quality clustering, the clustering quality was not a reliable predictor of the application performance. The best-performing clustering algorithm to date is the *k*-medoids.

How performing motion at different speeds can affect the search performance was studied in [25]. The first experiment featured a stretching of skeleton sequences by some fixed factor before their conversion into MWs. In the second experiment, all skeleton sequences were adjusted to have the same length. Either of these two experiments did not yield improved search performance. A different approach was then taken by producing multiple segmentations with different segment lengths. The segment shift was kept at 20% of each segment's length. The search was then issued multiple times, once for each segmentation. The results formed a candidate set, which was then used to refine the final answer. This approach yielded improved search performance.

There are several unresolved issues, such as indexing remains inefficient due to DTW. In the next chapter, we offer a solution to this problem and introduce a new quantization technique.

3 Composite Motion Word

This chapter features a novel approach to a definition of MW, using a principle that existing methods have yet not considered. Our definition is subsequently extended by relaxing the requirements for the type of MW elements used. The indexability of MWs is improved by analyzing the characteristics of MW sequences, and the MW sequence distance function is replaced with a suitable alternative. Finally, we propose an extension of this distance function to target specific settings where the distinction between categories with varying number of action repetitions is used.

3.1 Definition

The three previously presented MW approaches share the same minimal building block, a skeleton. In there, the whole skeleton is used to compare the similarity between two segments during clustering. We envision a different approach where we decompose the skeleton into natural body parts. We carefully design this new approach while preserving the favorable properties of MWs, namely comparable space requirements to the MWs created by the soft quantization.

We define *Composite Motion Word* (CMW) as a tuple $(p_1, p_2, \ldots, p_n)$, where p_i for $1 \leq i \leq n$ is a body part identifier and $n \in \mathbb{N}$ is fixed number of body parts the skeleton is divided into. Each body part is quantized using hard quantization, i.e., it is represented by a one-dimensional identifier (ID) – one hard MW.

The user specifies the number of body parts, but for the purposes of this work, we set n equal to five. The division of skeleton joints into body parts is shown in Figure 3.1. We use the following notation for each body part: the torso with the head is denoted as T, the right arm is denoted as RA, the left arm is denoted as LA, the right leg is denoted as RL, and finally the left leg is denoted as LL. We additionally define a set of all body parts $BP = \{T, RA, LA, RL, LL\}$, and a function $\mathcal{ID} : CMW \times BP \to \mathbb{N}$, which for a CMW and a body part returns the body part's ID from the CMW.

The essential part of every MW definition is the matching function. We first define when two body parts match using $match_p^{BP} : CMW \times$

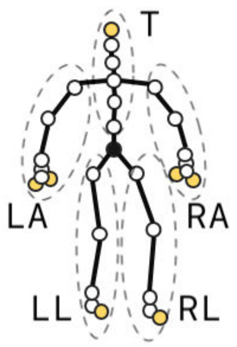


Figure 3.1: Division of a skeleton into body parts for CMW. Adapted with modifications from [7].

$CMW \to \{0,1\}$ function, where $p \in BP$. The function returns one when the IDs of the specified body part p are the same and zero otherwise. Formally as follows:

$$match_p^{BP}(A,B) = \begin{cases} 1 & \text{if } \mathcal{ID}(A,p) = \mathcal{ID}(B,p) \\ 0 & \text{otherwise} \end{cases}$$

Now we can define when two CMWs match using $match_m^{MW} : CMW \times CMW \to \{0,1\}$ function, where $m \in \mathbb{N}$ as follows:

$$match_m^{MW}(A,B) = \begin{cases} 1 & \text{if } \sum\limits_{p \in BP} match_p^{BP}(A,B) \geq m \\ 0 & \text{otherwise} \end{cases}$$

We define the matching function with respect to the parameter m. This definition allows us to flexibly set the appropriate number of body parts to match. Note that we do not impose any restrictions on which body parts should be matched. This stems from the fact that we may encounter a variety of actions, each of which requires us to match a different set of body parts.

Previously published MW quantization methods have focused on ways of tackling the border problem, namely soft and multi-overlay quantization. We address this issue implicitly by clustering each body part independently and allowing flexibility in the matching function.

Another motivation for flexibility is the following: if we say that all body parts should always match, we drastically limit the possible

CMW matches. Consider two people clapping their hands. The first person stands perfectly still on both legs, so the motion of, say, the right leg is assigned to some cluster. The second person steps forward with the right leg while performing the clapping. This movement causes the clustering process to assign the right leg body part to a different cluster. As a result, these two seemingly identical actions would not be the same in the eyes of the matching function for $m = 5$. We study the behavior of the parameter m in Subsection 5.2.1.

3.2 Joint Relations

The definition of the CMW allows us to partition the skeleton joints into non-overlapping sets, which are processed and matched separately. The division serves as the backbone of the CMW but may not capture all of the category nuances specific to particular joints. To increase the flexibility of the CMW, we incorporate a new type of MW element called a joint relation. A *joint relation* corresponds to an arbitrary set of joints and is created and processed the same way as a body part. It is intended to capture an interaction of specific joins that interact with each other. In addition, the relations can overlap and are added as new elements to the CMW. They are placed into defined positions in the CMWs.

3.3 Sequence Alignment of Motion Words

The original MWs were compared with DTW. The problematic nature of DTW lies in the absence of triangle inequality. The inequality does not hold, which hurts the indexability and makes kNN queries inefficient.

DTW is used twice throughout the MW creation. The first time is when calculating distances between skeleton segments. DTW also calculates the distance between two transformed MW sequences. We believe that DTW may not be able to exploit the specific characteristics of MWs, and therefore we advocate for an alternative. We have identified three fundamental properties that the algorithm for computing distance between two MW sequences should adhere to:

1. Be a metric. This property would greatly improve the indexability of MWs.

2. Respect the position of each MW in the sequence. The temporal part of the motion is encoded into relative positions of MWs, which precludes distances that treat the sequences as a set or compare transpositions of two adjacent objects.

3. Respect that the matching function has a Boolean output.

Considering these criteria, we observe that MW sequences exhibit string-like properties. The hard quantization assigns a one-dimensional identifier to every segment and defines a Boolean matching function. The concept remains the same for other quantization techniques. Although they produce more complex MWs from the point of view of the sequence distance algorithm, the input does not change. The matching function naturally maps to the distance between two characters, returning zero if the characters match and one if they do not.

Therefore, we propose to exploit this inherent similarity and use the edit distance [26], a string metric that satisfies the triangle inequality denoted as d_{edit}. We compare DTW and edit distance in Subsection 5.2.2.

3.4 Action Repetitions

During the analysis of the action classification task, we noticed an undesirable behavior. Actions with multiple repetitions are misclassified into categories with fewer action repetitions. The issue of misclassifications between categories with action repetitions is to an extent addressed in [27, 28, 29]. They proposed a modification of the dataset by merging some of the categories because, in their eyes, they have the same semantics. Our goal is to address this problem without the need for dataset modification. As a result, we define an extension of the MW sequence distance presented in the previous section. This extension is not strictly dependent on the CMW but serves as a general method for dealing with repetitions.

With this method, we target a specific setting where two actions differ in the number of occurrences. We assume that all the atomic actions that make up the action with repetitions are performed at roughly the same speed.

3.4.1 Extending Composite Motion Word with Length

One possible approach is to bound the length of the actions by a constant or some relative factor. Since the CMWs do not contain any information about the length of the action to which they belong, we would have to encode it. We could realize this by adding a new element into the CMW, which would contain, e.g., the original unprocessed length of the action. In order to come closer to the essence of MWs, we could cluster these lengths and replace the value with action length identifiers. This introduces the border problem, now relevant to the length of the action.

The problem with these solutions is that they require adding a new element to the MW. The easiest solution is to add the new element to each MW. Every MW in the same action would contain the same value, which considerably affects the space requirements of MWs. This would further make each MW no longer a standalone unit but somewhat dependent upon the action to which it belongs. Another solution would be to place the action's length only into the first MW of the action. Both approaches would have to rethink the matching at the sequence level and integrate the extraction and some comparison process directly into the sequence distance function.

3.4.2 Concatenation of Motion Word Sequences

We propose a solution that considers the space requirements of MWs. We demonstrate our idea with an example. Take two sequences $A = abc$ and $B = dabceabcf$, where $d_{edit}(A, B) = 6$. Focusing on the structure of the sequences, we could align the sequences if we were to duplicate A. Specifically, we could create a new action $A \cdot A$, which is twice as long as A and is created by appending the A to the end of A. Since we can align the sequences properly, we obtain $d_{edit}(A \cdot A, B) = 3$. We can observe that the sequence distance has decreased. If B did not contain a sequence similar to A in the second half, the distance would not decrease.

Based on this observation, we extend the MW sequence distance function as follows. Let A and B be two MW sequences such that $|A| \leq |B|$, where $|X|$ is a sequence length of the sequence X. We formalize our idea in a new sequence distance function $d_{repeated}$:

$$d_{repeated}(A, B) = \begin{cases} \infty & \text{if } d_{edit}(A \cdot A, B) < d_{edit}(A, B) \\ d_{edit}(A, B) & \text{otherwise} \end{cases}$$

We create a new artificial action based on the shorter action. We duplicate the shorter action A into a new two times longer action $A \cdot A$. Based on this action, we derive the above distance. It is based on the following idea. Suppose the new synthetic sequence is more similar to B than to the original action A. In this case, we say that the original sequences come from a different category, i.e., each has a different number of repetitions. We project this conclusion into the distance by returning ∞. This makes these sequences more distant and effectively removes them from the kNN query results, prioritizing different sequences. In the other case, if the distance is still the same or possibly increased, we return the edit distance $d_{edit}(A, B)$ on the original sequences.

3.4.3 Discussion

A notable feature of this distance is that it does not necessarily have to defeat the purpose of using the edit distance for the metric property. The implementation of this distance can be done as a post-processing filter. We can still use the edit distance while retrieving the nearest neighbors and then use the distance to filter the neighbors. The number of neighbors should be adjusted appropriately for the filtering step.

Finally, we would like to comment on the result of this method. Even though a shorter action may be two times shorter than a longer one, we cannot say that the shorter action is repeated only once and the longer one twice. The action may be performed more slowly.

4 Classification Framework

This chapter presents a two-stage classification framework based on CMWs. We elaborate on the addressed problems and provide details on the framework's design. Classification methods, a parameter tuning algorithm, and a two-stage classification are described. A commentary on the characteristics of the framework with suggestions for possible modifications concludes the chapter.

Even though a single classifier can detect motions across many categories, the more nuanced and body part-specific motions may not be recognized. One of the possible approaches is to implement a re-ranking mechanism. The re-ranking function would take the nearest neighbors as input and produce a finer-grained ranking. This method's main drawback is that the neighbors may not contain any action from the same category as the query action. Re-ranking cannot help in this situation. We aim to develop a different method that might allow us to increase the performance up to the theoretical maximum.

4.1 Motivation

We based our design on the following observations. The solution to the problem of *action repetitions* is realized by modifying the edit distance into the function called $d_{repeated}$. The distance requires non-trivial computational resources. Its use at the global level may hinder our goal of providing efficient indexability of MW sequences. We want to isolate its use and use it only when the performance benefit outweighs the incurred cost.

The *flexibility of the matching function* may only be suitable in some situations. The body parts that match between two successive CMWs may be different. If we can distinguish between two motions based on the right arm, we want to focus solely on this feature.

The *action locality* is the last issue not covered by the global classifier. Consider two categories: an object lying on a desk is either grabbed or deposited. In such a situation, the comparison mechanism should focus on the time interval when the actual object is being grabbed or deposited to identify the correct category.

Our idea is to refine the classification result by focusing on specific local needs, as presented above. We achieve this by introducing specialized classifiers. A specialized classifier makes decisions tailored to a reduced set of categories to achieve maximum classification performance. The framework makes use of CMWs and targets the problems of action repetition, the flexibility of the matching function, and action locality.

4.2 Architecture

The design of the framework consists of two stages. The first stage includes a single global kNN classifier that distinguishes all categories. The second stage includes several specialized kNN classifiers that target a pre-specified list of categories. The specialized classification decisions are made by matching specific body parts or comparing behavior in the surroundings of the action extremum. The main parameters of the specialized classifiers are tuned automatically.

The global classification is analogous to the classification in [1]. It features the same Weighted-Distance kNN Classifier (global classifier). We define a global classification function $\mathcal{G} : \mathcal{S} \to \mathcal{C}$ that, for an input sequence $s \in \mathcal{S}$ returns the classified category $C \in \mathcal{C}$. In our framework, we allow only a single global classifier.

A specialized classifier is also Weighted-Distance kNN Classifier. The specialized classification function is defined as $\mathcal{S}_{SC} : \mathcal{S} \to \mathcal{C}$, which for a sequence $s \in \mathcal{S}$ returns its classified category $C \in \mathcal{C}$ by the specialized classifier SC. We do not constrain the number of specialized classifiers in any way. Each specialized classifier is associated with an *invocation category*. The corresponding specialized classifier is invoked when the global classifier classifies an action into this category. The input to the specialized classifier is restricted in this sense, intercepting the global classification decision. It then makes its own classification decision based on a subset of categories, called *classification categories*. The main idea behind the specialized classification is that the specialized classifiers do not work over all dataset categories as the global classifier but rather over a smaller subsets.

We define two category-distinguishing methods which tackle observed issues and are implemented by the specialized classifiers. For

```
Algorithm 1: TrainSpecializedClassifiers($\mathcal{C}$, $t$)
Input: $\mathcal{C}$ – set of categories containing training actions, $t \in \mathbb{N}$ –
       maximal number of classification categories per specialized
       classifier
Output: $\mathcal{G}$ – global classification function, $SC_s$ – set of specialized
        classifiers
1 $A \leftarrow \{a \in C \mid C \in \mathcal{C}\}$
2 $GC \leftarrow$ create a new global classifier with global classification
    function $\mathcal{G}$ for a set of training actions $A$
3 $(G, m) \leftarrow$ ConstructMisclassificationGraph($\mathcal{C}$, $\mathcal{G}$)
4 $SC_s \leftarrow \emptyset$
5 foreach $(IC, CC) \in$ ExtractFromTransposedGraph($G, m, t$) do
6     $P \leftarrow$ TuneSpecializedClassifierParams($CC$)
7     $SC \leftarrow$ create a new specialized classifier based on parameters $P$
8     $SC_s \leftarrow SC_s \cup \{(IC, SC)\}$
9 return $(\mathcal{G}, SC_s)$
```

each set of categories, one or a combination of predefined methods is selected. The auto-tuning algorithm fully automates the process of creating specialized classifiers. After the initial tuning, the system is ready to receive actions for classification. Each action is then classified at most twice, once by the global classifier and then optionally by a specialized classifier. The creation, design, and tuning of the specialized classifiers is described in the following sections.

4.3 Classification Process

The training part of the two-stage classification process is captured in Algorithm 1. It takes a set of categories $\mathcal{C}$, where each training action is assigned to a single category, and a threshold t as input parameters. We create a global classifier to classify all training actions. Next, the misclassification graph is constructed. It is then given as input to the extraction mechanism, which produces pairs of invocation category IC and classification categories CC. The classification categories are transformed into specialized classifiers, while the association with the

Algorithm 2: TwoStageClassification($\mathcal{G}$, SC_s, A)

Input: $\mathcal{G}$ – global classification function, SC_s – set of specialized classifiers, A – action for classification

Output: category assigned to the action A

```
1 $C \leftarrow \mathcal{G}(A)$
2 foreach $(IC, SC) \in SC_s$ do
3     if $C = IC$ then
4         $C \leftarrow \mathcal{S}_{SC}(A)$
5 return $C$
```

invocation category is left unchanged. We obtained a single global classifier with a set of specialized classifiers.

The second part corresponds to the two-stage classification embodied by Algorithm 2. It starts by classifying the action A using the global classifier. It then iterates over the specialized classifiers, checking whether the category predicted by the global classifier is one of the invocation categories. If so, we invoke the associated specialized classifier, performing the second stage of the two-stage classification. When no specialized classifier is found, the result of the global classification is returned.

The following sections describe the inner workings of extracting invocation and classification categories, the category-discrimination methods used by the specialized classifiers, and the tuning process.

4.4 Representing Misclassifications

The first step in building a specialized classifier is to analyze the global classification result. In particular, we are interested in misclassified actions between categories and their frequency. These misclassifications are captured in a directed misclassification graph together with a misclassification function m. It acts as an edge-weighting function quantifying the number of misclassifications between two different categories.

Algorithm 3 takes a set of categories and a global classification function as input and constructs a directed misclassification graph

Algorithm 3: ConstructMisclassificationGraph($\mathcal{C}, \mathcal{G}$)

Input: $\mathcal{C}$ – set of categories, $\mathcal{G}$ – global classification function

Output: G – directed misclassification graph, m – misclassification function

```
1 Define $m : \mathcal{C} \times \mathcal{C} \to \mathbb{N}_0$, such that
    $\forall C, C' \in \mathcal{C} : m(C, C') = |\{a \in C \mid \mathcal{G}(a) = C'\}|$
2 $V \leftarrow \{C \in \mathcal{C} \mid \exists C' \in \mathcal{C} : C \neq C' \wedge (m(C, C') > 0 \vee m(C', C) > 0)\}$
3 $E \leftarrow \{(C, C') \in V \times V \mid C \neq C' \wedge m(C, C') > 0\}$
4 $G \leftarrow (V, E)$
5 return $(G, m)$
```

together with the misclassification function. The misclassification function captures the number of misclassifications between two different categories. Vertices correspond to categories from or to which at least one misclassification occurs. A directed edge from category A to category B is added when at least one misclassification occurs from A to B.

The misclassification function assigns weights to the edges of the graph according to the number of misclassifications. Note that only non-zero edges and no self-loops are present, hence the name misclassification graph. To use the constructed graph to correct the misclassifications, we need to transpose both the graph and the misclassification function. The idea is this. If an action is misclassified from category A to category B, we want to develop a method to reverse this misclassification. Our solution is to create a specialized classifier for actions from both categories and invoke it only when there is a classification into category B. If this is the case, we can analyze the action by the specialized classifier and classify it back to category A.

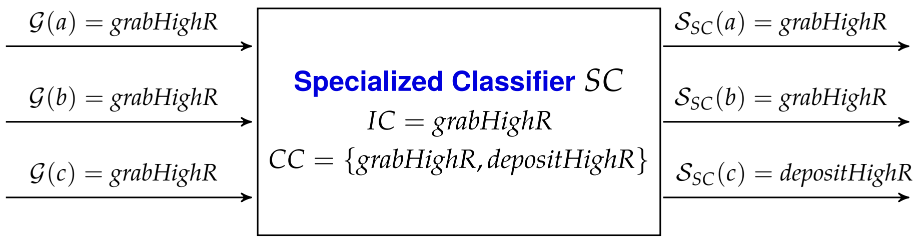
Algorithm 4 takes as input the set of categories, the directed misclassification graph, the misclassification function, and the parameter $t \in \mathbb{N}$ as a maximum number of classification categories per specialized classifier. As mentioned above, the transpositions of the graph and the misclassification function in the first two lines are responsible for reversing the misclassification decision. The algorithm iterates over the vertices of the transposed graph G^T. Outgoing neighborhood vertices are selected, and based on the cardinality of N_v, and

```
Algorithm 4: ExtractFromTransposedGraph($\mathcal{C}$, $G$, $m$, $t$)
Input: $\mathcal{C}$ – set of categories, $G$ – directed misclassification graph, $m$
       – misclassification function, $t \in \mathbb{N}$ – maximal number of
       classification categories per specialized classifier
Output: a set of ordered pairs $(IC, CC)$ where $IC$ is an invocation
        category and $CC$ is a set of classification categories
1 $G^T \leftarrow$ transpose graph $G$
2 Define $m^T : \mathcal{C} \times \mathcal{C} \to \mathbb{N}_0$, such that
   $\forall C, C' \in \mathcal{C} : m^T(C, C') = m(C', C)$
3 $R \leftarrow \emptyset$
4 foreach $v \in V(G^T)$ do
5     $N_v \leftarrow \{v' \in V \mid m^T(v, v') > 0\}$
6     $T \leftarrow$ take first $\min(t, |N_v|)$ categories in descending order from
       $N_v$ based on $m^T$
7     $R \leftarrow R \cup \{(v, T \cup \{v\})\}$
8 return $R$
```

the parameter t, the vertices corresponding to the most misclassified edges are retrieved. The current vertex v is placed first in an ordered pair, and the retrieved vertices T are placed second. These pairs are incrementally added to a result set that is returned.

The pairs returned by the algorithm serve as a basis for specialized classifiers. Each pair consists of an invocation category IC as the first element. Each invocation category is associated with one specialized classifier. This association defines when the specialized classifier is invoked, i.e., when the global classifier classifies an action into such an invocation category.

The second element of the pair is a set of categories CC called classification categories. A new specialized classifier is created based on the classification categories. It uses actions from these categories to make the classification decision and returns one of them as a result. An example of the relationship between the invocation category and the classification categories is shown in Figure 4.1.



$\mathcal{G}(a) = grabHighR$
$\mathcal{G}(b) = grabHighR$
$\mathcal{G}(c) = grabHighR$
Specialized Classifier SC
$IC = grabHighR$
$CC = \{grabHighR, depositHighR\}$
$\mathcal{S}_{SC}(a) = grabHighR$
$\mathcal{S}_{SC}(b) = grabHighR$
$\mathcal{S}_{SC}(c) = depositHighR$

Figure 4.1: An example of a specialized classifier created based on the result of the ExtractFromTransposedGraph algorithm, classifying actions a, b, and c. The classifier receives actions classified by the global classifier into the invocation category *grabHighR*. Each action is then classified into one of the classification categories in CC.

4.5 Selected Body Parts

Based on the two previously presented algorithms, we can define which categories a specialized classifier should be built on and when it should be invoked. A specialized classifier is tailored to the characteristic properties of a subset of categories. We use the following methods to distinguish between them: matching based on selected body parts of the CMW and matching a neighborhood around the extremum of the action.

In contrast to the existing quantization MW techniques, the CMW design allows us to attack the classification with a targeted approach. The global classifier uses a flexible matching function. It does not restrict which body parts should match, but requires a minimum number of matches. In specialized classification, we stick to comparing the same body parts throughout the sequence. The question is then which and how many body parts to choose. In the ideal case, it is a single body part that uniquely differentiates actions from two or more classification categories.

The following procedure we apply helps with identifying the best-performing body part and then adding another one. We do not search all possible combinations exhaustively. We call the best-performing body part a *primary* and the extra body part *secondary*. The counterintuitive fact is that we must use disjunction instead of conjunction to match the two body parts. Taking this process a few steps further, we could arrive at a complex formula that combines multiple body parts using conjunctions and disjunctions. We are aware of the danger of adding numerous body parts, so we limit the number of body parts to two. Algorithm 5 gives the pseudocode.

The algorithm captures the selection process of a single body part. The selection of both body parts and assessment of the viability of the repetition distance is implemented in Algorithm 11.

To match only selected body parts, we define a new matching function $match_S^{MW} : MW \times MW \to \{0,1\}$ for any $S \subseteq BP$. The function returns one if at least one of the body parts of the set S matches, and zero otherwise. We formally define this matching function as follows:

$$match_S^{MW}(A,B) = \begin{cases} 1 & \text{if } \sum\limits_{p \in S} match_p^{BP}(A,B) \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

In total, the specialized classifier SC_{BP} tuned by the SelectBodyPart algorithm uses four parameters $(CC$, bp_p, bp_s, and $r)$ to make a classification decision in addition to the classifier itself. The parameter r decides whether to use the action repetition distance. The classification process of the specialized classifier using body parts is then realized by Algorithm 6. The output of this algorithm is the result of $\mathcal{S}_{SC_{BP}}$.

4.6 Finding Extremum

While some actions can be distinguished by a movement specific to one or two body parts, some problematic actions still require an even more targeted approach. A typical example would be *grab* and *deposit* categories in [30]. Actions from these categories represent very similar motions. This leads to a tricky classification because the difference between the actions in a few MWs in the middle of a sequence may not be enough to categorize the action based on the MW distance. We could exploit the border problem if the actual grab and deposit motions were assigned to different clusters for a body part. Nevertheless, the problem persists as we are comparing the whole MW sequences.

The essence of such action is in a short period when the actual grabbing or depositing of the object occurs. Ideally, we would like to

Algorithm 5: SelectBodyPart(CC, BPs, bp_p, r)

```
Input: $CC$ – classification categories, $BPs$ – set of body parts to
       evaluate, optional parameter $bp_p$ – primary body part, $r$ –
       should the algorithm check for repeated actions
Output: configuration parameters for a specialized classifier
        together with the achieved accuracy
1  $A \leftarrow \{a \in C \mid C \in CC\}; R \leftarrow \emptyset$
2  $C_t \leftarrow$ create a temporary weighted-distance $k$NN classifier for the
   set of actions $A$
3  if $r$ is true then $d \leftarrow d_{repeated}$ else $d \leftarrow d_{edit}$
4  foreach $bp \in BPs$ do
5      if $bp_p$ is nil then $S \leftarrow \{bp\}$ else $S \leftarrow \{bp, bp_p\}$
6      $R_{bp} \leftarrow$ evaluate $C_t$, use $match_S^{MW}$ for matching CMWs and $d$
        for computing distance between two CMW sequences
7      if $bp_p$ is nil then
           ▷ Selected primary body part
8          $R \leftarrow R \cup \{(bp, \text{nil}, r, R_{bp}.accuracy)\}$
9      else
           ▷ Selected secondary body part
10         $R \leftarrow R \cup \{(bp_p, bp, r, R_{bp}.accuracy)\}$
11 return $\text{argmax}_{(bp_p, bp_s, r, accuracy) \in R}\ accuracy$
```

pinpoint the moment and its close neighborhood and compare actions based on these restricted surroundings. The method should not only be restricted by the neighborhood but also concern itself with only a specific body part where most semantics come from. To address this problem, we propose a new fine-grained approach. We aim to design an algorithm that automatically finds the right time, neighborhood, and body part. We incorporate all findings into a new distance.

Since skeleton partitioning and its subsequent transformation into CMWs provide a relatively coarse partitioning, we focus on the original actions comprising 3D coordinates of skeleton joints. We only apply the first step of the MW pipeline, the normalization, see Subsection 2.1.1, with frame rate reduction, see Subsection 2.1.2. To compute the distance between two skeleton actions, we use DTW algorithm.

```
Algorithm 6: BodyPartClassification($SC_{BP}$, $A$)
Input: $SC_{BP} = (CC, bp_p, bp_s, r)$ – specialized classifier
       configuration created based on the SelectBodyPart
       algorithm output, $A$ – action to be classified
Output: $\mathcal{S}_{SC_{BP}}(A)$ – category of action $A$
1 if $r$ is true then $d \leftarrow d_{repeated}$ else $d \leftarrow d_{edit}$
2 if $bp_s$ is nil then $S \leftarrow \{bp_p\}$ else $S \leftarrow \{bp_p, bp_s\}$
3 return classify action $A$, use $match_S^{MW}$ for matching CMWs and $d$
    for computing distance between two MW sequences
```

The computational requirements of DTW are minimized since just a specific body part, and its short motion is used.

We want to know which body part does the most movement. For this purpose, we use PCA [31] evaluated on a matrix of poses from all actions; see Figure 4.2. From the result, we extract the most significant axis a (a matrix column). This axis identifies a join j belonging to a body part. This body part is then used to calculate the distance within DTW.

As for selecting the right moment, we look for the minimum or maximum value in the axis throughout the action. To select the right neighborhood, we take a certain percentage based on the interval around the minimum or maximum value. Since we do not know which extremum to choose and what the correct percentage is, we have to tune both the parameters. The neighborhood is then created by taking a consecutive subsequence of poses around the pose corresponding to the chosen extremum.

Finally, to compare the two actions, we use the following procedure. We retrieve raw data of the normalized skeleton actions and construct a neighborhood for each action based on the parameters selected by the previous method. This neighborhood is then compared using d_{DTW} on the selected body part. The proposed method is realized by Algorithms 7–10.

$$M_{m\times n} = \begin{pmatrix} v_{1,1} & \cdots & \overbrace{v_{1,j} \quad v_{1,j+1} \quad v_{1,j+2}}^{\text{x, y, and z coordinate of } P^j} & \cdots & v_{1,n} \\ \vdots & & \vdots \quad\quad \vdots \quad\quad \vdots & & \vdots \\ v_{i,1} & \cdots & v_{i,j} \quad v_{i,j+1} \quad v_{i,j+2} & \cdots & v_{i,n} \\ \vdots & & \vdots \quad\quad \vdots \quad\quad \vdots & & \vdots \\ v_{m,1} & \cdots & v_{m,j} \quad v_{m,j+1} \quad v_{m,j+2} & \cdots & v_{m,n} \end{pmatrix} \begin{matrix} \\ \\ \}\ \text{pose } P_i \\ \\ \\ \end{matrix}$$

Figure 4.2: Matrix M used in the PCA computation in the SelectExtremumNeighborhood algorithm. The number of poses is denoted by m, the number of joint coordinates by n, P^j for $j \in \{1, 4, 7, \ldots, n-2\}$ represents xyz-coordinates of a single joint, and P_i for $i \in \{1, \ldots, m\}$ represents a pose.

4.7 Extremum Neighborhood

Algorithm 7 for classification categories CC, a Boolean parameter min, and a percentage p returns configuration parameters for extremum neighborhood specialized classifier. It starts by retrieving normalized skeleton actions from all classification categories. They are assembled into a matrix on which the PCA is initiated. The result is analyzed, and a column with the highest absolute value is extracted from the first principal component. This index contributed to the first component with the highest variance. Based on this index, the joint j, the axis a, and the corresponding body part bp are selected. The parameters are now tuned, and the algorithm computes a neighborhood distance. A temporary classifier is created and evaluated using the neighborhood distance. The configuration of the parameters is returned along with the accuracy of the classifier.

The extremum neighborhood classification is outlined in Algorithm 8. A neighborhood distance is created based on the result parameters of the SelectExtremumNeighborhood algorithm. It is then used to classify any given action A. The specialized classifier SC_{EN} using the extremum neighborhood method consists of the following

```
Algorithm 7: SelectExtremumNeighborhood($CC$, $min$, $p$)
Input: $CC$ – classification categories, $min$ – should we check
       neighborhood around minimum instead of maximum, $p$ –
       percentage
Output: configuration parameters for a specialized classifier
        together with the achieved accuracy
  ▷ Extraction of joint, axis, and body part from PCA
1  $A_o \leftarrow$ retrieve all normalized skeleton actions from $CC$
2  $P \leftarrow \{pose \in action \mid action \in A_o\}$
3  $M \leftarrow$ matrix where rows are poses from $P$ and columns are
     $xyz$-coordinates for each pose joint, see Figure 4.2
4  $R_{PCA} \leftarrow$ run PCA on $M$
5  $i \leftarrow$ retrieve the column index from the first principal component
     in $R_{PCA}$ with the maximal absolute value
6  $j \leftarrow \lfloor (i-1)/3 \rfloor$; $a \leftarrow (i-1) \bmod 3$
7  $bp \leftarrow$ the body part containing joint $j$
  ▷ Extremum neighborhood computation
8  $C_t \leftarrow$ create a temporary weighted-distance $k$NN classifier for a
     set of actions $A_o$
9  $d_{neighborhood} \leftarrow$ CreateNeighborhoodDistance($j, a, min, p, bp$)
10 $R_{bp} \leftarrow$ evaluate $C_t$, use $d_{neighborhood}$ for computing distance
     between two MW sequences
11 return $(j, a, min, p, bp, R_{bp}.accuracy)$
```

parameters besides the classifier: CC, j, a, min, p, and bp. The output of this algorithm is the result of $\mathcal{S}_{SC_{EN}}$.

Quantification of the distance between two input actions is done by Algorithm 9. It creates a $d_{neighborhood}$ distance custom for the passed input actions A and B. It internally retrieves the normalized raw skeleton actions and computes the neighborhood of each action. DTW is applied in the neighborhood on the body part bp. The returned distance is parameterized by joint j, axis a, Boolean min, percentage p, and body part bp. Each neighborhood is computed independently of the other and may result in a different number of poses in each action.

```
Algorithm 8: ExtremumNeighborhoodClassification($SC_{EN}$, $A$)
Input: $SC_{EN} = (CC, j, a, min, p, bp)$ – specialized classifier created
       based on SelectExtremumNeighborhood algorithm output,
       $A$ – action to be classified
Output: $\mathcal{S}_{SC_{EN}}(A)$ – category of action $A$
1 $d_{neighborhood} \leftarrow$ CreateNeighborhoodDistance($j$, $a$, $min$, $p$, $bp$)
2 return classify $A$, use $d_{neighborhood}$ for computing distance between
    two MW sequences
```

Finally, to select only the poses in the restricted surroundings of the extremum, Algorithm 10 extracts a consecutive subsequence of poses from the sequence of poses S, based on the joint j and the axis a. The algorithm has additional parameters, min, and p, where min indicates whether to search for a minimum or a maximum, and p is a percentage of the range of values based on which the neighborhood is later extracted. These two additional parameters are fine-tuned. The algorithm uses a function $\mathcal{V}$, which returns the device-estimated value for a pose-joint-axis combination.

4.8 Tuning Specialized Classifier

In the previous sections, we proposed two classification methods that specialized classifiers use to make category predictions. Now, we automate selecting the best method and its configuration.

Algorithm 11 takes classification categories CC as input and generates configuration parameters for either a body part or an extremum neighborhood classifier. It starts by selecting a primary body part from all available body parts BP. Then it selects a secondary body part that is different from the primary one. These body parts are additionally evaluated with $d_{repeated}$ by setting the last parameter of SelectBodyPart to true. The best-performing body part method configuration is then selected. If all actions are classified correctly, the procedure stops and returns this body part configuration. Otherwise, we have to use the SelectExtremumNeighborhood algorithm. We evaluate the algorithm over the predefined percentages and minimum and maxi-

```
Algorithm 9: CreateNeighborhoodDistance($j, a, min, p, bp$)
Input: $j$ – joint, $a$ – axis, $min$ – should we check neighborhood
       around minimum instead of maximum, $p$ – percentage, $bp$ –
       body part on which to compute the distance
Output: $d$ – distance function taking two MW sequences as input
        and computing $d_{DTW}$ distance on a consecutive
        sub-sequences on the body part $bp$
1 $d_{neighborhood} \leftarrow$ function $(A, B)$:
2     $A_o \leftarrow$ retrieve the normalized skeleton action $A$
3     $B_o \leftarrow$ retrieve the normalized skeleton action $B$
4     $N_A \leftarrow$ FilterNeighborhood($A_o, j, a, min, p$)
5     $N_B \leftarrow$ FilterNeighborhood($B_o, j, a, min, p$)
6     return $d_{DTW}(N_A, N_B)$ using joints from the body part $bp$
7 return $d_{neighborhood}$
```

mum. As in the previous method, we choose the best one. Finally, we compare the results of the methods and return the better one.

One may notice that we naturally prefer the result of SelectBodyPart. The classifier created by this method can make classification decisions based on the CMW alone, so it does not need any additional information like the original skeleton action.

4.9 Commentary

The computational requirements of the framework can be significantly adjusted. The user can select only the body part method with a single body part without the repetition distance. This configuration allows for increased classification and tuning speed since no flexible matching is required. Moreover, the particular body part can be extracted from the CMWs and stored directly as a one-dimensional identifier. On the other hand, one can utilize all of the presented methods. The methods should be easily adaptable since the main parameterization takes place in Algorithm 11.

The unsupervised nature of the global classifier constraints the overall performance. The specialized classifiers may be less effective if

Algorithm 10: FilterNeighborhood(S, j, a, min, p)

Input: S – sequence of poses, j – joint, a – axis, min – should we check neighborhood around minimum instead of maximum, p – percentage

Output: consecutive sub-sequence of poses from S containing only poses around the neighborhood of $P_{i_{min}}$ or $P_{i_{max}}$ based on the min parameter

```
1  $i_{max} \leftarrow \text{argmax}_{i \in \{1,\dots,|S|\}} \mathcal{V}(P_i, j, a)$
2  $i_{min} \leftarrow \text{argmin}_{i \in \{1,\dots,|S|\}} \mathcal{V}(P_i, j, a)$
3  $v_{max} \leftarrow \mathcal{V}(P_{i_{max}}, j, a); v_{min} \leftarrow \mathcal{V}(P_{i_{min}}, j, a); range \leftarrow |v_{max} - v_{min}|$
4  if $min$ is true then
5      $i \leftarrow i_{min}; t \leftarrow v_{min} + p * range$
6  else
7      $i \leftarrow i_{max}; t \leftarrow v_{max} - p * range$
8  $l \leftarrow i; r \leftarrow i$
9  while $l > 0$ and $\big((min$ is true and $\mathcal{V}(P_{l-1}, j, a) \leq t)$ or
10                   $(min$ is false and $\mathcal{V}(P_{l-1}, j, a) \geq t)\big)$ do
11     $l \leftarrow l - 1$
12 while $r < |S|$ and $\big((min$ is true and $\mathcal{V}(P_r, j, a) \leq t)$ or
13                   $(min$ is false and $\mathcal{V}(P_r, j, a) \geq t)\big)$ do
14     $r \leftarrow r + 1$
15 return consecutive sub-sequence of poses from $S$, such that $P_i \in S$
    for $i \in \{l, l+1, \dots, r-1\}$
```

the global classifier creates numerous misclassifications. They must distinguish between a larger number of actions and categories without the flexibility of the global classifier. It can be mitigated to some extent by the parameter t, restricting the maximal number of classification categories per specialized classifier.

The advantage of the two-stage design is that we can detach the specialized classifiers at any point. By doing so, we sacrifice the performance benefits of specialized classifiers to gain the possibility of unsupervised evaluation and increased classification speed.

Algorithm 11: TuneSpecializedClassifierParams(CC)

Input: CC – classification categories

Output: configuration parameters for a specialized classifier together with the achieved performance

```
1  $R_p \leftarrow$ SelectBodyPart($CC$, $BP$, nil, false)
2  $bp_p \leftarrow R_p.primaryBodyPart$
3  $R_s \leftarrow$ SelectBodyPart($CC$, $BP \setminus \{bp_p\}$, $bp_p$, false)
4  $bp_s \leftarrow R_s.secondaryBodyPart$
5  $R_p^R \leftarrow$ SelectBodyPart($CC$, $\{bp_p\}$, nil, true)
6  $R_s^R \leftarrow$ SelectBodyPart($CC$, $\{bp_s\}$, $bp_p$, true)
7  $B_{BP} \leftarrow$ the best-performing result based on the accuracy from
     $\{R_p, R_s, R_p^R, R_s^R\}$
8  if $B_{BP}$ achieved 100% then return $B_{BP}$
9  $P \leftarrow \{10\%, 20\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%\}$
10 $R_{EN} \leftarrow \emptyset$
11 foreach $p \in P$ do
12     foreach $min \in \{$true, false$\}$ do
13         $R_{EN} \leftarrow R_{EN} \cup \{$SelectExtremumNeighborhood($CC$, $min$,
             $p$)$\}$
14 $B_{EN} \leftarrow$ the best-performing result based on the accuracy in $R_{EN}$
15 if $B_{EN} > B_{BP}$ then return $B_{EN}$ else return $B_{BP}$
```

We allow flexibility in selecting which body parts to compare during the global classification. The specialized classifiers are more restricted in this regard. Each classifier has a defined way of making the classification decision, which benefits the explainability.

Finally, the design of our framework allows for a significant benefit. The framework can be extended later with new actions and categories. The phenomenon of catastrophic forgetting [32] present in neural networks does not affect our design. Removal of existing categories is also possible.

5 Evaluation

This chapter goes over parameter tuning and performance evaluation of the presented methods. The evaluation is done on two datasets acquired with different technologies. The appropriate parameter values are selected for the newly presented CMW. We study the proposed classification framework.

5.1 Datasets

5.1.1 HDM05

HDM05 [30] is a dataset of 3D skeleton sequences captured at 120 frames per second. It contains 2,345 actions grouped into 130 categories. The skeleton consists of 31 joints, tracked by the Vicon sensor. Five non-professional actors perform the actions.

For CMW, the actions are cut into 80-frame long segments shifted by 16 frames, resulting in over 28,000 segments in total (see [21, Table 4.4 and Figure 4.3] for the segmentation analysis). We reduce the frame rate to 12 frames per second. We use this dataset in two variations, HDM05-130, the original dataset, and HDM05-65 [27], a dataset with a reduced number of categories. In both cases, the number of neighbors taken into account by the classifiers is set to four, the same number as in the paper proposing motion words [1].

5.1.2 PKU-MMD

The PKU-MMD [33] dataset was introduced with an emphasis on multimodal action detection. It uses Kinect v2 to capture skeleton joints, red, green, and blue colors, depth, and infrared radiation. The skeleton consists of 3D locations of 25 skeleton joints. The dataset contains over 1000 motion sequences lasting approximately three to four minutes, captured at 30 frames per second. Each sequence contains approximately 20 actions, which are grouped into 51 action categories. We use 43 categories that consist exclusively of everyday activities. The other eight consist of human interactions. 66 different people perform actions in three camera views (left, middle, right). We consider 18 neighbors in the classifiers. This value is chosen to account for the increased number of actions per category compared to the previous dataset.

The dataset authors propose two evaluation settings: *Cross-View* (CV) and *Cross-Subject* (CS). The former evaluation targets the robustness of transformation, where left and right captured sequences are used for training and the middle sequences for testing. The latter divides the dataset based on subjects who perform the sequences. The training set consists of sequences performed by 57 subjects. Nine different subjects then perform the test set. The goal of this division is to be able to distinguish between movement variations within each category.

In the case of the PKU-MMD, the segments are 24 frames long and shifted by 4.8 frames. The number of segments is over 470,000. The frame rate is reduced from 30 to ten.

5.2 Composite Motion Word

This section is dedicated to finding suitable parameters for the CMW. Once selected, the process is repeated with the edit distance to ascertain whether the distance replacement for DTW has a meaningful impact. Finally, we evaluate and comment on the distance function for action repetitions.

5.2.1 Clustering Body Parts

In this subsection, we study the optimal parameter values of CMW while respecting both the clustering overhead and the application evaluation. We consider how many clusters each body part should be clustered into and the appropriate number of matches m in the matching function. For simplicity, we keep the number of clusters k the same for each body part. We study the classification and search performance concerning varying k and $m \in \{1, 2, 3, 4, 5\}$. The range of values of the parameter k is denoted on the x-axis in Figure 5.1.

Figure 5.1 shows the classification accuracy of CMW using the DTW algorithm. We observe the following trend: we get better accuracy with decreasing number of matches. It gradually increases

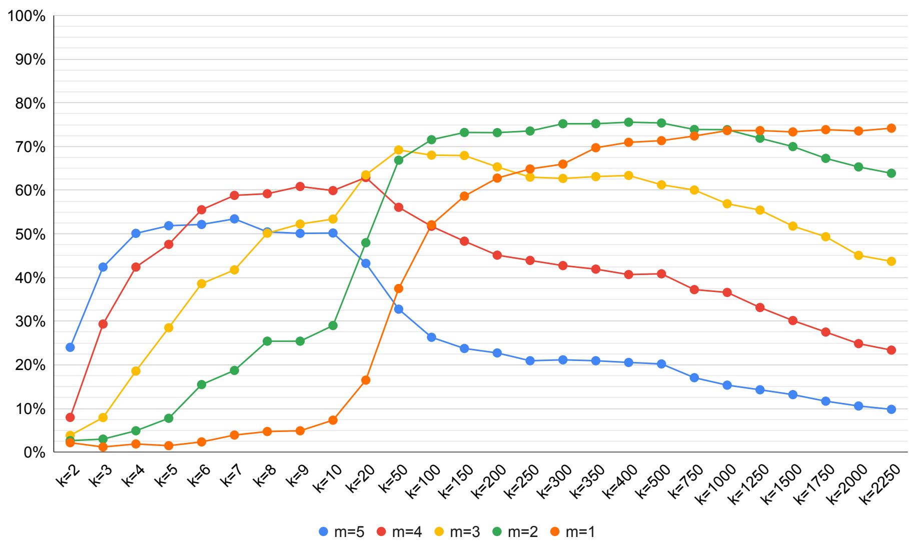


Figure 5.1: Classification accuracy of CMW on the HDM05-130 dataset with respect to parameters k and m using DTW distance to compare MW sequences.

from $m = 5$ as we decrease the number of matches and reaches its maximum value with $m = 2$, $k = 400$ at 75.57%. This confirms our earlier motivation for relaxing the number of required matches in Section 3.1. Although we could continue to evaluate results for higher ks, the significant computational requirements of clustering outweigh the minuscule application performance benefits.

Table 5.1 lists the best-performing search and classification configurations for each m. Parameter-wise, the optimal k and m were chosen to be 400 and 2 for classification tasks and 50 and 2 for search tasks. This is a good compromise between the resources required by the clustering and the achieved application performance. The selected configurations can outperform the hard quantization technique by 0.6 percentage points in the case of classification performance and 1.53 percentage points in the case of search performance. No single CMW configuration can outperform the existing MW quantization technique when both classification and search are considered simultaneously.

Table 5.1: Application performance of MWs on the HDM05-130 dataset with respect to parameters k and m using DTW to compare the MW sequences.

Type	m	k	Classification	Search
Composite	5	4	50.11%	30.16%
		7	53.43%	26.11%
	4	10	59.91%	37.99%
		20	62.90%	33.46%
	3	20	63.50%	43.05%
		50	69.21%	38.67%
	2	50	66.87%	**45.74%**
		400	**75.57%**	31.91%
	1	350	69.72%	43.27%
		1750	73.86%	26.32%
Hard [1]		350	74.97%	44.21%

We also tested more granular skeleton divisions with more body parts, but this approach yielded only a more computationally expensive clustering process. No significant benefits translated into the application performance.

5.2.2 Sequence Alignment of Composite Motion Words

This experiment aims to determine if the edit distance is a suitable replacement for DTW. We use the same evaluation as the previous experiment but replace d_{DTW} with d_{edit} when computing the distance between two MW sequences. We then study the changes in the application performance.

The classification accuracy of CMW with respect to parameters k and m when using d_{edit} is used is shown in Figure 5.2. It captures the increasing trend with increasing ms as in the case with DTW. The most notable increase in accuracy is achieved in single and double-digit ks.

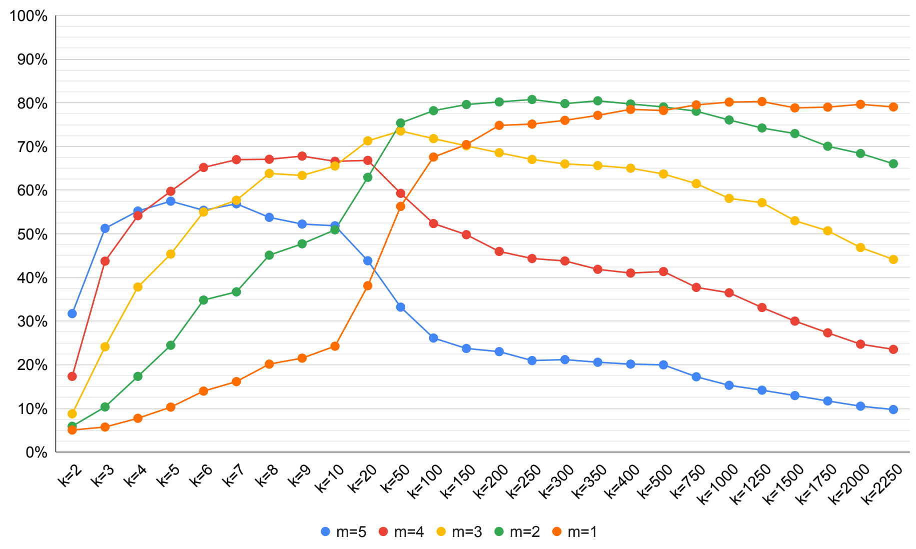


Figure 5.2: Classification accuracy of CMW on the HDM05-130 dataset with respect to parameters k and m using edit distance to compare MW sequences.

Regarding the parameter m, it shows the most significant increase in lower ms.

The best-performing configurations for each m are listed in Table 5.2. The classification accuracy using the edit distance with $k = 250$ reaches a maximum of 80.77%, an increase of 5.2 percentage points over the $k = 400$ configuration using DTW. Not only does the performance increase, but the maximum value peaks at the lower $k = 250$. This means that we have almost halved the number of clusters and we can increase the performance by changing the MW sequence distance function. In the case of the search task, the two best-performing configurations remain at the same value of $k = 50$. The search performance increases by 2.25 percentage points to 47.99%. The edit distance proves to be effective by not only increasing performance but also reducing the number of clusters needed.

Table 5.2: Comparison of the application performance of the best-performing CMWs and Hard MW on the HDM05-130 dataset with DTW and edit distance. m is equal to two.

	DTW			Edit distance		
	Composite		Hard	Composite		Hard
	$k = 50$	$k = 400$	$k = 350$	$k = 50$	$k = 250$	$k = 350$
Class.	66.87%	**75.57%**	74.97%	75.39%	**80.77%**	78.59%
Search	**45.74%**	31.91%	44.21%	**47.99%**	37.94%	43.31%

5.2.3 Action Repetitions

Table 5.3 lists the results and differences when d_{edit} is replaced by $d_{repeated}$. The classification performance increases to a maximum of 81.75%, an increase of almost one percent point. The best search performance decreases from a maximum of 47.99% to 47.04%. The classification accuracy of the best-performing configurations increases. The results confirm that it is beneficial to focus on action repetitions as suggested in Section 3.4. The classification process can take advantage of the substituted distance. In contrast, the search performance decreased in all configurations.

The design of the $d_{repeated}$ distance is independent of the type of MW used. In our case, the classification performance can be improved, but on the other hand, the performance gain does not outweigh the incurred computational cost. The method should be used in a specific setting where the number of actions is limited, and we know that using this distance can improve the result. Therefore, we do not use this distance globally.

5.3 Classification Framework

In this section, we analyze the misclassifications made by the global classifier. Then, we study the performance impact of each of the presented methods. We evaluate the framework using a dataset with fewer

Table 5.3: The achieved application performance of $d_{repeated}$ distance compared to d_{edit} on the best-performing configurations. Evaluated on the HDM05-130 dataset.

		$m = 5$	$m = 4$	$m = 3$	$m = 2$	$m = 1$
Classification	Accuracy	59.36%	68.87%	75.69%	**81.75%**	81.02%
	Change	+1.88%	+1.07%	+2.13%	+0.98%	+0.72%
Search	Accuracy	30.79%	38.56%	42.06%	**47.04%**	46.04%
	Change	-0.45%	-0.50%	-0.81%	-0.95%	-0.60%

categories and compare it to neural networks. Finally, we conclude with an experiment employing the PKU-MMD dataset.

We use the following notation. $|SC|$ denotes the number of specialized classifiers generated by the framework. This count includes the number of classifiers using the body part method, denoted as $|SC_{BP}|$, and the extremum method, denoted as $|SC_{EN}|$. The accuracy of the specialized classifiers, abbreviated as *SC accuracy*, is the number of actions assigned to the correct category by the specialized classifiers divided by the number of actions classified by the specialized classifiers.

5.3.1 Analysis of Classification Performance

In the previous experiments, we were able to significantly increase the classification performance of the CMW by adopting the edit distance. A follow-up question may be as to why the performance did not increase further. More specifically, we would like to know if there is a systematic problem that we could potentially mitigate, or if we have reached the limits of the current classification method. We analyze the classification result of the best-performing CMW configuration $k = 250$, $m = 2$ using the edit distance.

The category with the highest number of misclassified actions is *grabHighR* with 21 misclassifications, followed by *depositHighR* with 19. Here, we make three crucial observations. First, in both cases, there is a category into which most actions are misclassified. The misclassifications are not uniformly distributed; a majority is classified into a single category. Second, both categories are misclassified from

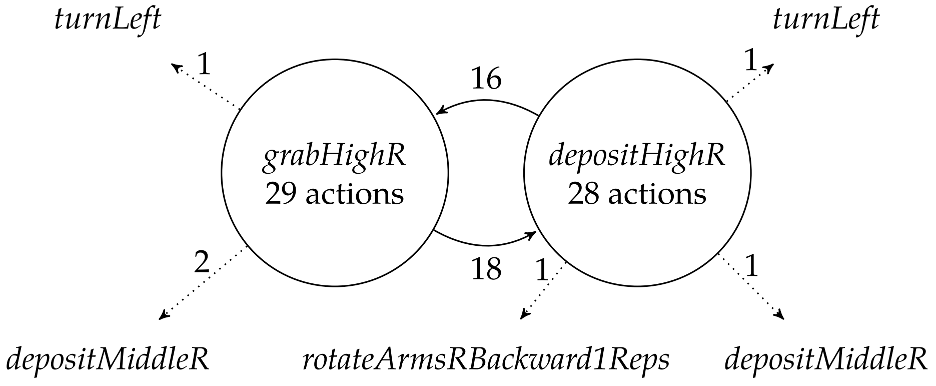


Figure 5.3: A part of the misclassification graph for categories *grabHighR* and *depositHighR* evaluated on the HDM05-130 dataset. 18 actions in *grabHighR* are classified as *depositHighR*. 16 actions in *depositHighR* are classified as *grabHighR*. The rest are misclassifications into other categories.

one to the other. Finally, these categories are very close in terms of general movement. This misclassification relationship is depicted in Figure 5.3.

After analyzing all categories, we observed several recurring misclassification tendencies. We summarize them in the following list:

1. Actions from a category with multiple action repetitions are classified into a category with fewer repetitions. The repetitions also take the form of hops or steps (e.g., *kickRSide2Reps* to *kickRSide1Reps*).

2. Actions are misclassified between categories with a similar movement (e.g., *walk2StepsLstart* and *shuffle2StepsLStart*).

3. Actions from categories where the motion is made in a specific direction on the spot (e.g., *grabHighR* and *depositHighR*; *kickLFront1Reps* and *kickLSide1Reps*).

These observations led us to the definition of the classification framework, which serves as a correction mechanism for the misclassifications made by the global classifier. The $d_{repeated}$ distance addresses the first item. The second and third items are addressed implicitly by restricting the matching function to predefined body parts. In specialized classifiers, all three points are the essence of the body part method. The extremum neighborhood method was specifically designed to address the particular situation of grabbing and depositing. It can be thought of as a solution to the coarse nature of the MW.

5.3.2 Performance Impact of Classification Methods

Since we are using labels within the classification framework, we need to change the evaluation scenario. Instead of using the leave-one-out approach, we use n-fold cross-validation. The dataset is split into n non-overlapping sets, respecting the number of actions in each category. The following process is repeated for each $i = 1, \ldots, n$. The $n-1$ sets, different from i, form the training samples used in the run i. The actions from set i are test samples for measuring the application performance. The resulting performance is then averaged over the n runs.

We conduct experiments under seven different configurations. To account for all misclassifications, the experiments are evaluated with $t = \infty$, the maximum number of classification categories per specialized classifier. The first configuration, *Without Specialized Classifiers*, evaluates the CMW in the cross-validation settings using only the global classifier. The other configurations are implemented as modifications of Algorithm 11 (page 33) by omitting some internal results from consideration before returning the best-performing one. The experiments are evaluated using the two-stage classification framework.

The *Primary BP* configuration uses specialized classifiers but is limited to no extremum neighborhood classifiers and only a single body part (R_s, R_p^R, R_s^R, and R_{EN} omitted). The following configurations, *Secondary BP with AND* and *Secondary BP with OR*, extend the previous one by allowing a secondary body part. The first one is matched in a hypothetical situation where we always require the match of both body parts (R_p^R, R_s^R, and R_{EN} omitted). The second corresponds to the definition of the matching function for specialized body part classifiers (R_p^R, R_s^R, and R_{EN} omitted). *Action Repetitions* is the last configuration studying the effects of body part classification methods. It builds upon

Table 5.4: Classification accuracy of CMW and the two-stage classification framework with the listed modifications. Evaluated using 2-fold cross-validation.

	SC accuracy	Accuracy	St. dev.
Without Specialized Classifiers		71.77%	0.72
Primary BP	71.30%	74.41%	0.71
Secondary BP with AND	71.20%	74.33%	0.95
Secondary BP with OR	73.63%	76.42%	0.20
Action Repetitions	74.18%	76.89%	0.38
Only EN	72.89%	75.78%	0.83
Combination BPs + EN	75.95%	**78.38%**	1.36

the *Secondary BP with OR* configuration by using the action repetition distance in performance-improving situations (R_{EN} omitted).

To provide insight into the performance of the extremum neighborhood method, we use a configuration called *Only EN*. It uses the extremum neighborhood classifiers and does not allow body part classifiers (R_p, R_s, R_p^R, and R_s^R omitted). Finally, the *Combination BPs + EN* configuration does not restrict Algorithm 11 in any way. Both extremum neighborhood and body part specialized classifiers are used. The body part classifiers use the same configuration as *Action Repetitions*.

Table 5.4 lists the obtained results from the evaluation. Note that results in this table are evaluated using 2-fold cross-validation, while the previous experiments used a leave-one-out approach. Adding specialized classifiers in the restricted settings defined by the *Primary BP* configuration improves the accuracy by more than three percentage points over the global configuration. An interesting observation can be made by comparing the *Primary BP* and *Secondary BP with AND* configurations. They produce almost identical results, suggesting that a disjunction should be used when matching two body parts, as captured by the *Secondary BP with OR* configuration. The *Action Repetitions* configuration is the best-performing configuration using only body part classifiers, improving the accuracy by over five percentage points compared to global classification.

The *Only EN* configuration studies the situation where we exclusively use the extremum neighborhood method. On the one hand, it can outperform the single body part evaluation. On the other hand, the configuration is outperformed when the secondary body part is added to the body part classifiers. The last configuration, *Combination BPs + EN*, proved to be the best-performing one, producing 59.5 body part classifiers and 38.5 extremum neighborhood classifiers on average. Specialized classifiers classified 85.42% of the test set actions on average.

5.3.3 Merging Similar Categories

In [27], the authors argue that some categories of the HDM05-130 dataset are identical and should be merged. As an example, they state that actions from the categories *walk2steps* and *walk4steps* should be treated as actions belonging to a single category *walk*. They combine several categories of the original HDM05-130 dataset according to the following rules:

- actions that differ in the number of repetitions performed are combined into a single category;
- actions that differ in the starting limb are combined into a single category.

Based on these rules, a new dataset named HDM05-65 is created with 2,345 actions divided into 65 categories. They evaluate the dataset using 10-fold cross-validation. The evaluation results of different classification methods [27, 28, 29] in conjunction with our approach are listed in Table 5.5.

Our technique achieves 90.02% accuracy, less than six percentage points below a neural network approach [27]. In addition, on average, the two-stage classification produced 47.6 specialized classifiers (14.1 body part, 33.5 extremum neighborhood), while the classification accuracy of the specialized classifiers reached 88.18%. On average, the specialized classifiers classify 82.95% of the actions from the test set. The standard deviation of the CMW and the framework was 2.2 and 0.98, respectively.

We see the dataset modification as a partial solution to our problems listed in Section 5.3.1. The rules suggested in [27] essentially

Table 5.5: Classification accuracy of 10-fold cross-validation on the HDM05-65 dataset. Composite Motion Word and Two-Stage Classification framework evaluated with parameters $k = 250, t = \infty$.

Method	Accuracy
Composite Motion Word	85.98%
Two-Stage Classification	90.02%
Multi-Layer Perceptron [27]	95.59%
Hierarchical Bidirectional RNN [28]	96.92%
Deep LSTM + Co-occurrence [29]	97.25%

target one of our observed problems: misclassifications between categories with different numbers of action repetitions.

The authors of [28] list the categories *grabHighR*, *depositHighR*, *grabMiddleR*, *depositMiddleR*, *grabLowR*, and *depositLowR* as the main misclassifications in their approach. This phenomenon corresponds to the third item of our list of misclassification tendencies. In our case, differentiating between such motions is a matter of favorable clustering. The actions are not well-separated. We can only reasonably distinguish between them using a more sophisticated method, such as the proposed extremum neighborhood.

5.3.4 Joint Relations

The second dataset, PKU-MMD, is captured with less accurate sensors. The size of the dataset is almost an order of magnitude larger than the HDM05 dataset. Some of the categories feature motions requiring precise capturing in order to be correctly differentiated. The imprecise nature of PKU-MMD motivates the introduction of joint relations. Upon closer inspection, many of the categories exhibit motion with a hand around the head, e.g., *saluting*, *brushing hair*, *touching the head*, or *wearing glasses*. Along with the imprecision, this makes it difficult to classify the data solely on the body part composition of the CMW.

To stay within the limits of the CMW representation, we define three joint relations. They consist of a set of joints captured separately: right hand (RH), left hand (LH), and head (H). They are subse-

Table 5.6: Evaluation comparison of CMW and two-stage classification with and without the three relations in the CS and CV settings of the PKU-MMD dataset.

		Relations	$\|SC\|$	$\|SC_{BP}\|$	$\|SC_{EN}\|$	SC accuracy	Accuracy
CS	CMW	✗					48.35%
		✓					62.26%
	Two-Stage	✗	43	23	20	52.49%	52.49%
		✓	43	28	15	64.77%	64.77%
	BiLSTM [34]						84.73%
	Activity images + CNN [35]						85.00%
CV	CMW	✗					65.48%
		✓					75.73%
	Two-Stage	✗	43	22	21	68.46%	68.46%
		✓	43	27	16	77.01%	77.01%
	Activity images + CNN [35]						92.00%
	BiLSTM [34]						92.11%

quently combined to form the relations of $RH + LH$, a relation of the right hand and the left hand, $RH + H$, a relation of the right hand and the head, and $LH + H$, a relation of the left hand and the head. They are processed in the same way as body parts and form new elements in CMW. They are used in the same situations as the body parts; they are used in the classification process on both global and specialized levels.

Experiments are carried out with the number of clusters equal to 300. We set the maximum number of classification categories per specialized classifier (parameter t) to two. In contrast to the previous evaluations, the training and test sets are defined for the PKU-MMD.

Table 5.6 shows the increased classification performance when the relations are added. A substantial increase can be seen for both the CMW and the two-stage classification framework. In the context of the two-stage classification, the added relations increase the accuracy by more than 12 percentage points in CS and over eight percentage points in the CV settings. This change also positively affects the number of classifiers using body parts. The exact values of the two-stage classification accuracy and the accuracy of the specialized classifiers reflect the fact that every action was subject to the specialized classification.

The performance benefit of specialized classifiers is limited, achieving only a slight increase in accuracy. There are numerous misclassifications on the global level due to the imprecise nature of the dataset and the size of each category. The other possibility is that the unsupervised core of the framework cannot contend with neural networks. In previous experiments, we always used $t = \infty$ to account for all misclassifications. In this case, we have to limit it to two to achieve the best accuracy since it tends to decrease for larger t.

6 Conclusion

This thesis presented a new type of motion word (MW) called composite motion word (CMW). We considered a skeleton divided into five body parts, each clustered into the same number of clusters. The evaluation showed that the optimal number of clusters for the classification task is 400. The accuracy reached 75.57% outperforming the hard quantization. The same division featuring 50 clusters per body part was used to reach a search performance of 45.74%, again surpassing the hard motion word. These results were achieved when the number of matches in the matching function was equal to 2.

The edit distance successfully replaced DTW for calculating the distance between two MW sequences. In the case of the CMW, the classification accuracy increased by over five percentage points to 80.77%. This change significantly increased the indexing capabilities of MWs. It also reduced the number of clusters required by nearly half to 250. As a result of this improvement, the CMW in this configuration outperforms all originally presented methods in classification accuracy. Finally, the action repetition distance, motivated by the numerous repetition-based categories in the HDM05 dataset, further improved the classification accuracy to 81.75%.

The definition of the CMW inspired the two-stage classification framework. After observing the results of the classification task, we developed a correction mechanism designed to address misclassifications called specialized classifiers. Focusing on subsets of categories, the specialized classifiers are designed to target the problems of action repetition, the flexibility of the CMW matching function, and action locality.

Three misclassification tendencies were observed and mapped to the presented classification methods. Seven different configurations were examined, providing insight into the advantages of the proposed methods. The accuracy improved from the base 71.77% to 78.38% when utilizing all of the presented methods.

The subsequent experiment compared existing neural network solutions to our proposal. On the modified dataset, the framework achieved an accuracy of 90.02%, less than six percentage points below the solution using a multi-layer perceptron. Although unsupervised at its core, the framework provides a semi-explainable and dynamic solution with a reasonable trade-off in classification accuracy.

Finally, the size and capturing precision of the PKU-MMD dataset motivated the introduction of joint relations. The nearly an order-of-magnitude larger dataset, captured with off-the-shelf hardware, limited the accuracy improvement of the classification framework. Nevertheless, the relations were successful and provided a significant improvement in accuracy. The accuracy in the cross-subject settings increased by over 12 percentage points to 64.77%. The cross-view settings achieved an increase of over eight percentage points to 77.01%.

Future research could investigate the effect of a different number of clusters for each CMW body part. A new mechanism limiting the number of actions processed by the specialized classifiers should also be considered. Systematically, the mechanism should allow more actions to be classified only by the global classifier without needing specialized ones.

A Attachments

The attached archive contains the following.

- `code` – Source code.
  - `clustering` – Scripts for splitting the dataset, running segment clustering, converting actions into motion words, and combining multiple hard motion words into a single composite motion word.
  - `motionvocabulary` – Converts actions into motion words. Extended with joint filter. Modifications are marked with the corresponding author tag in JavaDoc.
  - `mcdr` – Implementation of composite motion word and its sequence distances. Implementation of the tuning process, its methods, and specialized classifiers. Evaluation of the application performance in the cross-validation setting. Modifications are marked with the corresponding author tag in JavaDoc.
- `data` – Composite motion word actions with accompanying files used for evaluation.

Bibliography

1. SEDMIDUBSKY, Jan; BUDIKOVA, Petra; DOHNAL, Vlastislav; ZEZULA, Pavel. Motion Words: A Text-Like Representation of 3D Skeleton Sequences. In: *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part I*. Lisbon, Portugal: Springer-Verlag, 2020, pp. 527–541. ISBN 978-3-030-45438-8. Available from DOI: `10.1007/978-3-030-45439-5_35`.
2. BATKO, Michal; NOVAK, David; ZEZULA, Pavel. MESSIF: Metric Similarity Search Implementation Framework. In: *Proceedings of the 1st International Conference on Digital Libraries: Research and Development*. Pisa, Italy: Springer-Verlag, 2007, pp. 1–10. DELOS'07. ISBN 3540770879.
3. SEDMIDUBSKY, Jan; ELIAS, Petr; BUDIKOVA, Petra; ZEZULA, Pavel. Content-Based Management of Human Motion Data: Survey and Challenges. *IEEE Access*. 2021, vol. 9, pp. 64241–64255. ISSN 2169-3536. Available from DOI: `10.1109/ACCESS.2021.3075766`.
4. ELEFTHERIADIS, A.; JACQUIN, A. Chapter 1 - Image and Video Segmentation. In: *Advanced Video Coding: Principles and Techniques*. Elsevier, 1999, vol. 7, pp. 1–68. Advances in Image Communication. ISSN 0928-1479. Available from DOI: `https://doi.org/10.1016/S0928-1479(99)80003-5`.
5. MEHTA, Dushyant; SOTNYCHENKO, Oleksandr; MUELLER, Franziska; XU, Weipeng; ELGHARIB, Mohamed; FUA, Pascal; SEIDEL, Hans-Peter; RHODIN, Helge; PONS-MOLL, Gerard; THEOBALT, Christian. XNect: Real-Time Multi-Person 3D Motion Capture with a Single RGB Camera. *ACM Trans. Graph.* 2020, vol. 39, no. 4. ISSN 0730-0301. Available from DOI: `10.1145/3386569.3392410`.
6. DESMARAIS, Yann; MOTTET, Denis; SLANGEN, Pierre; MONTESINOS, Philippe. A review of 3D human pose estimation algorithms for markerless motion capture. *Computer Vision and Image Understanding*. 2021, vol. 212, p. 103275. ISSN 1077-3142. Available from DOI: `https://doi.org/10.1016/j.cviu.2021.103275`.

7. *Demonstration Application: Motion Data Processing* [online]. DISA, 2013 [visited on 2023-02-09]. Available from: `http://disa.fi.muni.cz/demo/motion-retrieval/`.

8. BUDIKOVA, Petra; KLEPAC, Daniel; RUSNAK, David; SLOVAK, Milan. Visual Exploration Of Human Motion Data. In: *Similarity Search and Applications: 15th International Conference, SISAP 2022, Bologna, Italy, October 5–7, 2022, Proceedings*. Bologna, Italy: Springer-Verlag, 2022, pp. 64–71. ISBN 978-3-031-17848-1. Available from DOI: `10.1007/978-3-031-17849-8_6`.

9. SHAIKH, Muhammad Bilal; CHAI, Douglas. RGB-D Data-Based Action Recognition: A Review. *Sensors*. 2021, vol. 21, no. 12. ISSN 1424-8220. Available from DOI: `10.3390/s21124246`.

10. REN, Bin; LIU, Mengyuan; DING, Runwei; LIU, Hong. *A Survey on 3D Skeleton-Based Action Recognition Using Learning Method*. arXiv, 2020. Available from DOI: `10.48550/ARXIV.2002.05907`.

11. VAHDANI, Elahe; TIAN, Yingli. Deep Learning-based Action Detection in Untrimmed Videos: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2022, pp. 1–20. Available from DOI: `10.1109/TPAMI.2022.3193611`.

12. HU, Xuejiao; DAI, Jingzhao; LI, Ming; PENG, Chenglei; LI, Yang; DU, Sidan. Online human action detection and anticipation in videos: A survey. *Neurocomputing*. 2022, vol. 491, pp. 395–413. ISSN 0925-2312. Available from DOI: `https://doi.org/10.1016/j.neucom.2022.03.069`.

13. SEDMIDUBSKY, Jan; ELIAS, Petr; ZEZULA, Pavel. Effective and Efficient Similarity Searching in Motion Capture Data. *Multimedia Tools and Applications*. 2018, vol. 77, no. 10, pp. 12073–12094. ISSN 1380-7501. Available from DOI: `10.1007/s11042-017-4859-7`.

14. ARISTIDOU, Andreas; COHEN-OR, Daniel; HODGINS, Jessica K.; CHRYSANTHOU, Yiorgos; SHAMIR, Ariel. Deep Motifs and Motion Signatures. *ACM Trans. Graph.* 2018, vol. 37, no. 6. ISSN 0730-0301. Available from DOI: `10.1145/3272127.3275038`.

15. MÜLLER, Meinard. Dynamic Time Warping. In: *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 69–84. ISBN 978-3-540-74048-3. Available from DOI: 10.1007/978-3-540-74048-3_4.

16. KAUFMAN, Leonard; ROUSSEEUW, Peter J. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 2005. ISBN 0-471-73578-7.

17. LLOYD, S. Least squares quantization in PCM. *IEEE Transactions on Information Theory*. 1982, vol. 28, no. 2, pp. 129–137. Available from DOI: 10.1109/TIT.1982.1056489.

18. ZEZULA, Pavel; AMATO, Giuseppe; DOHNAL, Vlastislav; BATKO, Michal. *Similarity Search: The Metric Space Approach*. Vol. 32. Springer, 2006. Advances in Database Systems. ISBN 0-387-29146-6.

19. SEDMIDUBSKY, Jan; ELIAS, Petr; ZEZULA, Pavel. Enhancing Effectiveness of Descriptors for Searching and Recognition in Motion Capture Data. In: *2017 IEEE International Symposium on Multimedia (ISM)*. 2017, pp. 240–243. Available from DOI: 10.1109/ISM.2017.39.

20. SEDMIDUBSKY, Jan; ZEZULA, Pavel. Probabilistic Classification of Skeleton Sequences. In: *Database and Expert Systems Applications*. Cham: Springer International Publishing, 2018, pp. 50–65. ISBN 978-3-319-98812-2. Available from DOI: 10.1007/978-3-319-98812-2_4.

21. MARTINČÍK, Tomáš. *Application of clustering algorithms for motion data quantization*. Brno, 2020. Available also from: https://is.muni.cz/th/w6gl2/. Bachelor's thesis. Masaryk University, Faculty of Informatics. Supervised by Petra BUDÍKOVÁ.

22. ESTER, Martin; KRIEGEL, Hans-Peter; SANDER, Jörg; XU, Xiaowei. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland, Oregon: AAAI Press, 1996, pp. 226–231. KDD'96.

23. CAMPELLO, Ricardo J. G. B.; MOULAVI, Davoud; SANDER, Joerg. Density-Based Clustering Based on Hierarchical Density Estimates. In: *Advances in Knowledge Discovery and Data Mining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN 978-3-642-37456-2.

24. JARVIS, R. A.; PATRICK, E. A. Clustering Using a Similarity Measure Based on Shared Near Neighbors. *IEEE Trans. Comput.* 1973, vol. 22, no. 11, pp. 1025–1034. ISSN 0018-9340. Available from DOI: `10.1109/T-C.1973.223640`.

25. BAGAR, Matěj. *Similarity-based Matching of Fast and Slow Motions using Motion Words*. Brno, 2022. Available also from: `https://is.muni.cz/th/o3ujj/`. Bachelor's thesis. Masarykova univerzita, Fakulta informatiky. Supervised by Petra BUDÍKOVÁ.

26. LEVENSHTEIN, Vladimir I. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*. 1965, vol. 1, pp. 8–17.

27. CHO, Kyunghyun; CHEN, Xi. Classifying and visualizing motion capture sequences using deep neural networks. In: *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*. 2014, vol. 2, pp. 122–130.

28. DU, Yong; WANG, Wei; WANG, Liang. Hierarchical recurrent neural network for skeleton based action recognition. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1110–1118. Available from DOI: `10.1109/CVPR.2015.7298714`.

29. ZHU, Wentao; LAN, Cuiling; XING, Junliang; ZENG, Wenjun; LI, Yanghao; SHEN, Li; XIE, Xiaohui. Co-Occurrence Feature Learning for Skeleton Based Action Recognition Using Regularized Deep LSTM Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2016, vol. 30, no. 1. Available from DOI: `10.1609/aaai.v30i1.10451`.

30. MÜLLER, M.; RÖDER, T.; CLAUSEN, M.; EBERHARDT, B.; KRÜGER, B.; WEBER, A. *Documentation Mocap Database HDM05*. 2007-06. Tech. rep., CG-2007-2. Universität Bonn. ISSN 1610-8892.

31. PEARSON, Karl. On Lines and Planes of Closest Fit to Systems of Points in Space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. 1901, vol. 2, no. 11, pp. 559–572. Available from DOI: `10.1080/14786440109462720`.

32. FRENCH, Robert M. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*. 1999, vol. 3, no. 4, pp. 128–135. ISSN 1364-6613. Available from DOI: `https://doi.org/10.1016/S1364-6613(99)01294-2`.

33. LIU, Chunhui; HU, Yueyu; LI, Yanghao; SONG, Sijie; LIU, Jiaying. PKU-MMD: A Large Scale Benchmark for Skeleton-Based Human Action Understanding. In: *Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities*. Mountain View, California, USA: Association for Computing Machinery, 2017, pp. 1–8. VSCC '17. ISBN 9781450355063. Available from DOI: `10.1145/3132734.3132739`.

34. ELIAS, Petr; SEDMIDUBSKY, Jan; ZEZULA, Pavel. Understanding the Limits of 2D Skeletons for Action Recognition. *Multimedia Syst.* 2021, vol. 27, no. 3, pp. 547–561. ISSN 0942-4962. Available from DOI: `10.1007/s00530-021-00754-0`.

35. VERNIKOS, Ioannis; KOUTRINTZES, Dimitrios; MATHE, Eirini; SPYROU, Evaggelos; MYLONAS, Phivos. Early Fusion of Visual Representations of Skeletal Data for Human Activity Recognition. In: *Proceedings of the 12th Hellenic Conference on Artificial Intelligence*. Corfu, Greece: Association for Computing Machinery, 2022. SETN '22. ISBN 9781450395977. Available from DOI: `10.1145/3549737.3549786`.