Masaryk University
Faculty of Informatics

Information Retrieval Techniques for 3D Human Motion Data

Master's Thesis

Bc. Ján Horváth

Brno, Fall 2020

Masaryk University
Faculty of Informatics

Information Retrieval Techniques for 3D Human Motion Data

Master's Thesis

Bc. Ján Horváth

Brno, Fall 2020

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. Ján Horváth

Advisor: prof. Ing. Pavel Zezula, CSc.
Consultant: RNDr. Jan Sedmidubský, Ph.D.

Acknowledgements

I would like to thank Doctor Sedmidubský, Doctor Budíková and Professor Zezula for including me in their research. Working with them was an interesting and enlightening experience.

Abstract

Recently, there has been an advance in pose estimation methods which would allow an efficient extraction of human poses from ordinary videos. Therefore, it is reasonable to expect an explosion in availability of motion capture data. Most motion-processing research is targeted at short motion sequences called actions or motion data streams. The thesis, made in collaboration with the DISA laboratory, focuses on medium length motion sequences called *episodes* and their similarity-based matching. We transform the data into a text-like representation and combine it with mature text retrieval techniques. The aim of the thesis is to propose an unsupervised two-phase episode matching model which surpasses the naive sequence alignment approach in both effectiveness and efficiency.

Keywords

motion capture, episode, episode matching, motion words, term frequency, inverted document frequency, sequence alignment, cosine similarity, Jaccard coefficient, bag-of-words, shingling

Contents

Introduction **1**

1 Episode matching **3**
1.1 *Motion capture data format* 3
1.2 *Related research* 3
1.3 *Problem definition* 6
1.4 *Dataset* 6
1.5 *Measuring precision and efficiency* 7

2 Baseline distance function **9**

3 Using motion words for episode matching **11**
3.1 *Hard motion words* 11
3.2 *Multi-overlay motion words* 13
3.3 *Episode matching with motion words* 13

4 Text retrieval methods for hard motion word processing **15**
4.1 *Indexability* 15
4.2 *Set and bag representations of hard motion words* 15
4.3 *Jaccard coefficient and cosine similarity* 16
4.4 *Introduction of weights – IDF* 17
4.5 *Introduction of weights – TFIDF* 18
4.6 *Experiment results* 18

5 Using multi-overlay motion words to improve precision **21**
5.1 *No weights* 21
5.2 *Adding weights* 22
5.3 *Experiment results* 23

6 Considering order in episodes **27**
6.1 *Episode matching using OF scenarios* 27
6.2 *Episode matching using OS scenarios* 29

7 Two-phase episode matching **31**
7.1 *Filtering and refinement* 31
7.2 *Order-free matching* 31

7.3 *Order-sensitive matching* 32
7.4 *Matching on the original dataset* 34

8 Shingling **37**
8.1 *Shingling for sequences of hard motion words* 37
8.2 *Experiment results* 38

9 Conclusion **41**

Bibliography **43**

Introduction

Motion capture has been used by filmmakers and game development studios to create lifelike animations of characters since the beginning of the twenty-first century. Examples of some of the first fully digital and well known characters created using motion capture are Jar Jar Binks from Star Wars or Gollum from the Lord of the Rings trilogy [1]. In the game development industry, we can see the importance of motion capture in games such as L.A. Noire, in which the player needs to make decisions based on the facial expressions and body language of characters. This gameplay mechanic required detailed and lifelike characters who are easier to capture than to model from scratch.

Motion capture data have traditionally been captured by specialized hardware, such as the relatively cheap Microsoft Kinect or high-end multi camera setup with reflective markers. Recently, however, there have also been advances in pose-estimation methods which allow extracting the motion capture data from ordinary videos [2] [3] [4] [5]. Because of that, it is reasonable to expect that there will be an explosion of widely available motion capture data.

Much like in the case of other data formats such as a text or an image, in order to make use of the data we first need to be able to process it, store it in a database and efficiently retrieve similar data. We already have mature techniques that can do it all for texts and images, but for motion capture data this is still a work in progress. There have been many published techniques which focus on modelling similarity among short (few seconds) motion capture sequences known as actions. Most of them employ a neural network in the process [6] [7] [8]. However, there is hardly any research that deals with the task of determining similarity between motion sequences taking a few minutes, such as figure skating performances or walking patterns. Such task presents additional challenges which do not arise in the case of action similarity searching. In this thesis, I present the research done in collaboration with members of Laboratory of Data Intensive Systems and Applications (DISA). The research has also been published in the proceedings of the 22nd IEEE International Symposium on Multimedia [9]. My main contribution is the experimental evaluation of the approaches described in this thesis. In our research, we denote a

motion episode to be a motion sequence lasting from tens of seconds to minutes and explore the problem of episode similarity searching.

In Chapter 1, I discuss the related research and describe the problem of *episode matching*, the dataset and the general experiment that we are using. In Chapter 2, we employ the dynamic time warping algorithm as our naive baseline solution. In Chapter 3, we improve on that by converting the data to a more efficient representation. In the next two Chapters (4 and 5), we improve our solution further by combining the representation with mature text-retrieval techniques. In Chapter 6 and 7, we discuss the order-free and order-sensitive variations of the episode matching problem and show how the two-phase model can be used for these variations. In Chapter 8, I mention another method which naturally follows from the text-retrieval approach. However, the results strictly worsen so I do not follow up on this method. Finally, Chapter 9 concludes the results and considers possible improvements.

We believe that there are many areas where the efficient processing, storing and searching of motion capture data would be desirable. In medical science, it can be used to assess treatment for patients with movement disabilities or to monitor their recovery. In sports, it could be used to evaluate figure skating or dancing performances. Additionally, it is also crucial for human-computer interaction and virtual reality.

1 Episode matching

We will start by explaining what motion capture data is and present a range of commonly researched motion capture tasks. We will then define the episode matching task and put it into the broader context of motion capture problems. We will also mention the dataset that we will be working with and the setup of our general experiment.

1.1 Motion capture data format

The term *motion capture*, or *MoCap* in short, is used to describe the process of recording movement of objects or, more commonly, people. MoCap data are created by tracking joints on a human body over time. Individual captures of whole human body are often referred to as *skeletons* or *poses*. Therefore, we can look at MoCap data as *sequences of skeletons/poses*. Each pose consists of body joints tracked in two or three dimensional space. Therefore, each joint is represented by two or three floating point numbers. For example, a one minute recording with 120 Hz frame rate with 30 tracked joints in three dimensions contains 648 000 floating point numbers, which is about 2.5 MB.

Human MoCap data are often categorized by their length. *Gestures* are very short sequences, usually shorter than a second, expressing an idea or a feeling, such as raising a finger or nodding your head. *Actions* are sequences, lasting a few seconds, which include movement of a larger part of your body, for example, standing up or opening a door. *Long motions* are general sequences possibly taking even several hours. These can include any lengthy recordings such as MoCap of patient's movements throughout the day or athlete's performance during a match.

1.2 Related research

The most researched MoCap tasks relate to actions. *Action recognition* deals with the problem of classifying an action into one of preselected classes. This is a task commonly solved by neural networks. In [10] the authors convert skeleton actions into 2D RGB images effectively

transforming the task into an image classification problem which they solve using four models: support vector machines, k-nearest neighbours, random forests and convolutional networks (CNN). In all the datasets they have used, the CNNs proved most accurate with the recognition accuracy ranging from 74% to 92%. Ke *et al.* [7] have also made use of the image action representation and train a local CNN along with a latent global network. They average the class prediction score of both of these networks which gives them superior results compared to individual predictions of any one of them. They have even managed to correctly predict almost 64% of the actions by seeing only the first 40% poses of each action. Avoiding skeleton sequence datasets, Ahmad and Khan [6] also used CNNs to combine black and white front images of actors performing actions captured by Kinect with data from inertial sensors. They have also augmented the images by rotation, flipping and adding noise to artificially generate more data.

Graph convolutional networks (GCN) are also used for the action recognition task due to their strengths in preserving skeleton structural information as seen in [11] or [12]. For example, Liu *et al.* [11] have added a hidden conditional random field model which predicts the class label based on the local and global features extracted by a GCN. Their so called GCN-HCRF model manages to preserve the spatial and temporal information to achieve state-of-the-art results on three action detection datasets.

Besides CNNs and GCNs, the long short-term memory (LSTM) recurrent networks are also a viable option since they are generally well suited for sequential and temporal data. In [13], the authors enhance an LSTM network solution with the introduction of a global context memory cell. It helps the two LSTM layers to selectively focus on the more informative joints. The cell functions similarly to the other gates in the LSTM cell such as forget gate. In [14], Wu *et al.* combine deep spatiotemporal features extracted using their R-LSTM network with a wavelet temporal pattern to define an action descriptor. The pattern is created from an action divided into subactions using a clustering tree built from bottom to top.

Action detection (and recognition) extends the previous task by including the problem of correctly identifying the beginnings and endings of actions in a longer stream of MoCap data. In [15] the authors

use a sliding window technique to select potential actions of varying length. The motion features of the action, extracted by a deep convolutional neural network, are then annotated with an action profile label if their similarity to the profile surpasses a threshold. Another approach is to use LSTM networks which are able to detect actions in offline streams or even in real time a few hundreds milliseconds ahead [16].

Another task, similar to action detection, is called *subsequence search*. It deals with detecting a short query sequence in a longer MoCap sequences or even streams. The difference between this task and the previous one is that the short query does not need to be an action with some semantic meaning. Any short sequence is viable for searching. The subsequence search usually involves partitioning of the long MoCap sequence into shorter overlapping subsequences to which the query can be compared. In [17] the query sequence is also partitioned into non-overlapping segments. The authors use a measure based on feature vectors extracted by a neural network to find matching subsequences for each individual query segment. These subsequences are then merged together, respecting the order of the query segments, and one candidate of the same length as the initial query is selected in each merged sequence. The candidates are selected using a joint-coordinate similarity measure and are ranked using the previously mentioned feature vector based distance measure. This approach achieves higher precision and search efficiency with much lower long motion segmentation compared to earlier papers. Another issue that comes up in the subsequence search problem is the speed variability at which the short motion is performed compared to its match in the long motion. This is addressed in [18]. Compared to [17] the authors partition the long motion several times into so called levels, each time with a different segment size. The multi-level segmentation supports speed-dependent retrieval of a query by only checking the segmentation level with segment sizes closest to the query size and also supports speed-invariant retrieval by also checking some of the surrounding levels. The amount of levels and the maximum ratio difference between the query size and the closest segment size is controlled by setting a *covering factor* parameter beforehand.

In this work, we focus on a new category of motions called *episodes* and their matching. An *episode* is a sequence of human actions which

takes anywhere from tens of seconds to several minutes. It has naturally defined semantics. Some examples include an actor's performance, an athlete's performance such as figure skating, workout routines and so on. In contrast to the action detection and recognition tasks, we present an approach which does not require labeled data. We do not use any trainable models but we rather focus on finding a good representation for the data and finding a capable similarity function which can quantify how similar two episodes are.

1.3 Problem definition

The aim of *episode matching* is to find semantically similar episodes by identifying common patterns in movements. For example, in the case of workout routines, we want to detect push ups or crunches.

We formally define *episode* to be a sequence $S = (P_1, P_2, \ldots, P_n)$ of n consecutive 3D skeleton *poses*. Each *pose* consists of k tracked *joints* in time. Each *joint* consists of xyz-coordinates represented by three floating point numbers.

The formal definition of the *episode matching problem* is to find a *distance* (*dissimilarity*) function

$$\mathcal{D}_S : S \times S \to \mathbb{R}^+$$

which allows us to find semantically similar k-nearest neighbours (sequences) for any given query (sequence) or to do range queries. We will compare several distance function and discuss their precision and efficiency.

1.4 Dataset

We looked through several popular motion capture datasets with sufficiently long motion sequences, for example, PKU-MMD [19], LSMB19 [20] and HDM05 [21]. We decided to use the HDM05 dataset which consists of 324 sequences (episodes) in total and tracks 31 body joints. We decided against including sequences which were poorly or vaguely annotated or were made up of too few actions. We ended up with 241 sequences. The dataset uses *scenarios* to semantically distinguish the

1.	walk 5 steps
2.	turn around (right)
3.	walk 5 steps (ducked)
4.	walk 5 steps (backwards)
5.	walk 5 steps sideways to the right, feet crossing over alternately front/back
6.	...

Figure 1.1: An incomplete example of a scenario from the HDM05 dataset. A scenario consists of several consecutive actions which are performed by an actor.

episodes. This fact is at the core of our experimental evaluation of the distance functions. A *scenario* is simply a list of actions that the actor performed during the motion capture. An example can be seen in 1.1. Therefore, each episode is a MoCap of an actor performing a particular scenario. On average, there are about 15 episodes per scenario. All scenarios combined define 130 different actions. It should be noted that there is one scenario which has only one episode associated with it. We will call it *singular episode* and we will treat it differently when carrying out the experiments.

1.5 Measuring precision and efficiency

Our experiment is designed to measure two aspects of the distance function: *precision* and *efficiency*.

Precision is measured using k-nearest neighbours search (kNN) with fixed or *variable* k. We denote the *variable* k value as k^*. kNNs are computed for each of the 240 sequences, omitting the *singular episode*. Note that even though we do not compute kNN for the *singular episode* itself, it can still be a part of a result of any other kNN query.

$$Prec = \frac{\sum_{i=1}^{240} \frac{c_i}{k_i}}{240}$$

where

- k_i is the number of sequences which share their associated scenario with sequence i. This is the k^*, i.e. variable k, value. (In the case of fixed k, the k_i value is always k)

- c_i is the number of sequences which share their associated scenario with sequence i and were contained in kNN search result for the sequence i

Efficiency is defined simply as the time required to evaluate all 240 queries including any necessary preprocessing. We evaluate *efficiency* on the same hardware every time.

2 Baseline distance function

As stated in 1.3, our task is to find a distance function which best reflects dissimilarity between two sequences of poses (*episodes*). In this chapter, we select a distance function, call it a *baseline distance function* and run the experiment on the sequences of poses. We also refer to this experiment as *baseline experiment*. In the following chapters, we attempt to improve this result by changing the distance function and the data representation. The baseline experiment gives us a reference result to which we can then compare the other experiments.

Since we are working with temporal sequences, we first try a sequence alignment method, the dynamic time warping (DTW)[22], as our *baseline distance function*. In general, DTW algorithm computes the *optimal matching* of points from one sequence to the other. There are several conditions that the DTW algorithm follows:

- First points of the sequences are always matched together;
- Last points of the sequences are always matched together;
- Every point must have at least one match in the other sequence
- If points of indices a and b are matched then there must not be a matching of indices A and B such that $a < A$ and $b > B$. In the Figure 2.1 this condition means that the dashed lines, which represent the matches, do not cross.

Optimal matching is a matching that satisfies these conditions and the sum of distances between all pairs of matched points is the smallest. In our case, the points of these sequences are poses. The distance between two poses is computed as a sum of distances between corresponding joints:

$$\mathcal{D}_P(P_1, P_2) = \sum_{j=1}^{31} \left\| P_1^j - P_2^j \right\| \tag{2.1}$$

The precision of DTW algorithm is 74.84%, which is reasonable. However, there are three problems that we need to address.

1. *Computation time*
   The DTW algorithm has quadratic time complexity and the pose

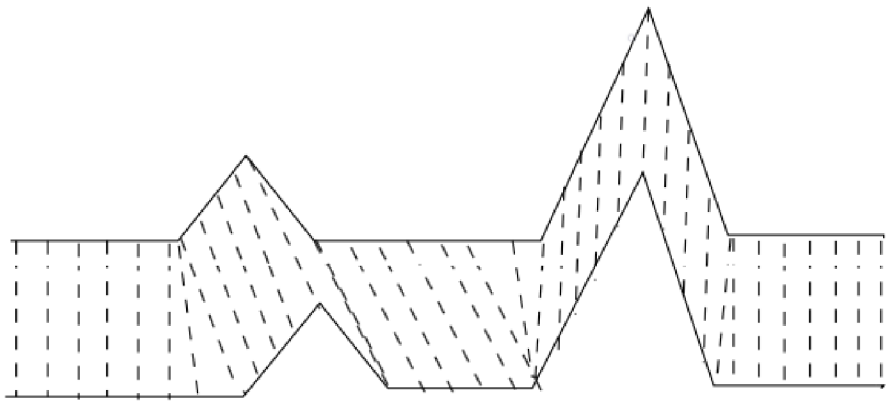


Figure 2.1: An example of what the resulting DTW matching looks like. The dashed lines represent matchings. Image taken from [23].

distance computation can take quite a long time for such a great number of poses. It took short of 25 minutes to evaluate all 240 queries in our experiment which is about six seconds per query. That is unusable for larger datasets.

2. *Indexability*
   Indexing techniques are commonly used in databases to avoid expensive distance computation by potentially ruling out large amounts of data which cannot be part of the result. Therefore, we want to be able to use some form of indexing. However, our data are too complex to be indexable. Same goes for the DTW distance measure because it does not respect the triangle inequality.

3. *Order-free episode matching*
   Imagine that we wanted to match episodes without taking the order of actions into account. If we were to compare two identical episodes, DTW would assign zero distance. However, if we took one of the episodes and shuffled its order of actions or switched its first and second halves, the DTW distance would increase considerably. In these cases, the DTW would fail since it relies heavily on the order of poses in sequences.

3 Using motion words for episode matching

Let us start with decreasing the computation time by simplifying the representation. The original MoCap data, although very accurate, can be quite unwieldy. In order to make the computation faster, we are transforming the data into a more manageable format. Recently, two papers have been published which recommend partitioning longer motions into smaller segments, thereby reducing dimensionality. One approach [24] is to transform the segments into frequency distribution of motion motifs called motion signatures. However, we are using the other technique described in [25]. This technique transforms the original data into a text-like representation so that we can make use of text-retrieval methods. Individual characters in the text-like representation are called *motion words* (MWs).

The process of *motion words* creation is shown in Figure 3.1. Each *motion word* corresponds to a segment of the original data. Each segment consists of m consecutive skeleton poses. We can look at each segment as a high-dimensional vector or rather a point in a high-dimensional space. These vectors are used as an input for the k-means algorithm, which separates the vector space into non-overlapping partitions. Each segment is then assigned a *motion word* (a natural number) based on the partition in which it is contained. The segments are overlapping and they are taken using a sliding window. In our case, the window size, i.e. the segment size, is 80 poses and we slide the window by 16 poses every time, therefore two consecutive segments have 80% overlap.

In addition to reducing the amount of data thanks to the transformation, we also make the distance computation much simpler. Instead of computing distance between joints as in 2.1, we use Boolean valued matching function for *motion words*:

$$\mathcal{M} : MW \times MW \to \{0, 1\} \tag{3.1}$$

3.1 Hard motion words

Hard motion words (HMWs) are created just as described above, meaning every motion word is a single natural number. The number is

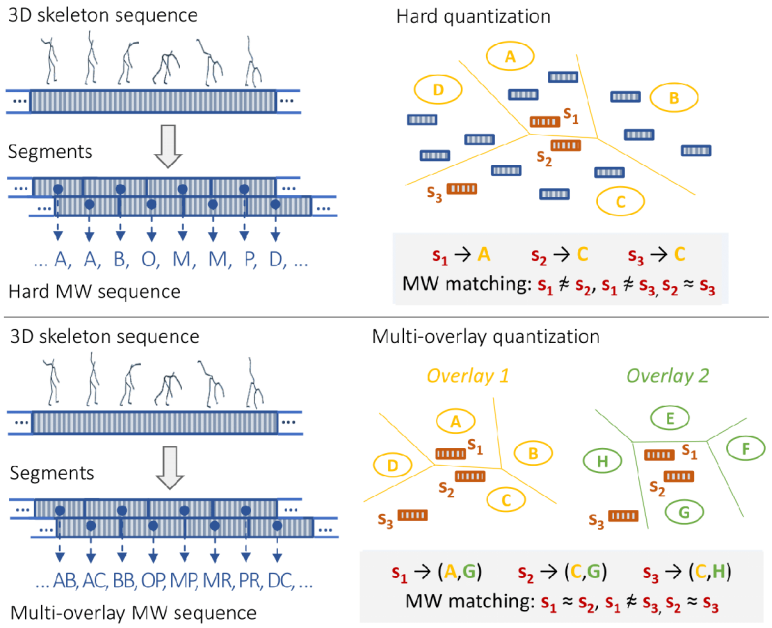


Figure 3.1: The construction of hard (top) and multi-overlay (bottom) motion words from sequences of poses, i.e. the original data. The image is a composition of images from [25].

assigned based on the k-means partition. Their advantage is that their matching function simply compares the two numbers and returns 1 if and only if the numbers are the same. The disadvantage of *hard motion words* is that they suffer considerably from so-called *border problem*. The border problem occurs when two segments that are close to each other but contained in different partitions are assigned different motion-words. This is an undesirable effect. An example occurrence of the problem can be seen in picture 3.1 for the segments s_1 and s_2.

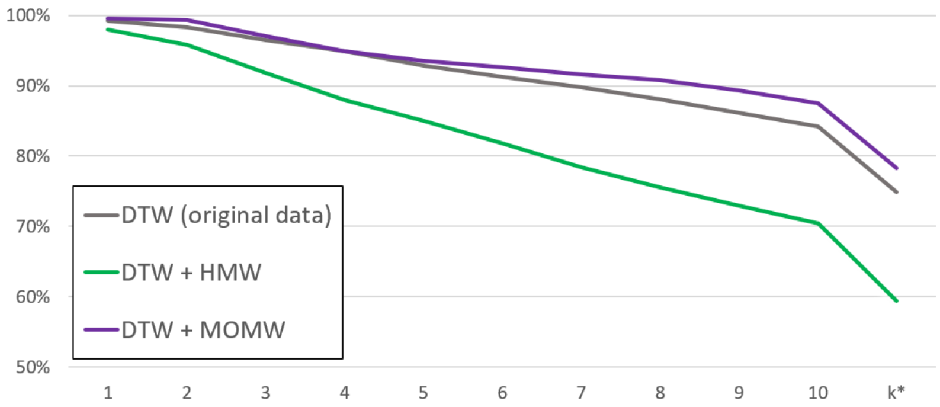
3.2 Multi-overlay motion words

Multi-overlay motion words (MOMWs) are created by repeating the process used to create HMWs multiple times. Each time a new random k-means partitioning is generated and the segments are assigned one number for every such partitioning. Because of that, it is highly unlikely for the border problem to occur in every partitioning for any two close segments. Again, notice segments s_1 and s_2 in 3.1. The *multi-overlay motion words* are therefore defined as a r-tuples of natural numbers where r is the number of partitionings. The matching function takes a parameter s such that $s \leq r$, compares the r-tuples and returns 1 if and only if the two tuples match at s positions or more. We set $s = 1$ unless specified otherwise. Such matching function for MOMWs takes longer than in the case of hard motion words but it gives us better results.

3.3 Episode matching with motion words

In order to address the problem of slow episode matching computation, let us run our experiment again and change the data representation from sequences of poses to sequences of motion words. We try both, the hard as well as the multi-overlay motion words. We keep the DTW algorithm as the distance function which means that the indexability issue still persists. We evaluate the kNN with $k = \{1, 2, \ldots, 10\}$ and $k = k^*$.

The efficiency is around 40–50 times higher than baseline efficiency reducing the computation time for a single query from 6 seconds to around 0.12 second (for HMWs) and 0.15 second (for MOMWs) in the



	k^* precision	Efficiency
Baseline	74.84%	25 min
DTW + MOMWs	78.28%	37 s
DTW + HMWs	59.35%	28 s

Figure 3.2: Precision graph of the experiment with DTW as the distance function for $k = \{1, \ldots, 10, k^*\}$. Comparison of MW representation of the episodes versus the original representation (sequences of poses).

case of $k = k^*$. While we do lose some precision in the case of HMWs, surprisingly, the MOMWs perform slightly better than the baseline. This demonstrates the effect of the border problem that HMWs suffer from. The results are shown in Figure 3.2.

4 Text retrieval methods for hard motion word processing

In this chapter, we are going to address the order-free matching issue as well as the quadratic time complexity of DTW and discuss the indexability issue. When we represent episodes as sequences of motion words we can see that they have similar properties to text. For example, in the case of HMW representation, the episodes are simple sequences of natural numbers which could be directly interpreted as text. Therefore, we will use some text retrieval techniques.

4.1 Indexability

We have to discuss indexability for HMWs and MOMWs separately. Sequence of HMWs is basically a text. Therefore, we can index the episodes as documents for example using the inverted index, which is commonly used in text retrieval. In our case the inverted index could map motion words to their positions in episodes speeding up the retrieval of episodes requested by a query.

On the other hand, compared to HMWs, the MOMWs are not scalars so the same indexing cannot be used. The problem is that while the boolean matching function (3.1) is an equivalence relation for HMWs, it is not an equivalence relation for MOMWs. It violates transitivity, for example, for MOMWs $w_1 = (11, 20, 31, 41, 51)$, $w_2 = (10, 20, 30, 40, 50)$, $w_3 = (12, 22, 30, 42, 52)$. The words w_1, w_2 are matching on their second position, the words w_2 and w_3 are matching on their third position but w_1 and w_3 are not matching. For this reason, the indexing techniques do not work well for MOMWs.

4.2 Set and bag representations of hard motion words

We start by changing the sequence representation of HMWs into a *set-of-HMWs* and a *bag-of-HMWs*. Neither of them keeps the information about the original order. The difference between them is that the bag representation preserves the count of individual HMWs while the set ignores it. The *bag-of-HMWs* is essentially a multiset. This simplifies

the representation but we can no longer use the DTW distance function with sets and bags since it requires the MWs to be ordered as a sequence. Notice that by shuffling the motion words in an episode, we get a different sequence of motion words but as soon as we transform the sequence to a set (or a multiset) we get the same result as if we transformed the original unshuffled sequence. In theory, this should work for order-free but not for order-sensitive episode matching. We will examine this in more detail in Chapter 6.

4.3 Jaccard coefficient and cosine similarity

A common way to compute a similarity between sets is the *Jaccard similarity coefficient*. We will modify the Jaccard similarity coefficient for the bag representation to get a suitable distance function. For *sets-of-HMWs* S_1, S_2 the distance function is simply one minus Jaccard coefficient:

$$\mathcal{D}_J(S_1, S_2) = 1 - \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \tag{4.1}$$

For *bags-of-HMWs* $B_1 = \{(w_1, n_1), \ldots (w_b, n_b)\}, B_2 = \{(w_1, m_1), \ldots (w_b, m_b)\}$, where w_i are motion words and m_i, n_i are motion word multiplicities for the word w_i, the notion of union and intersection needs to be modified:

$$\mathcal{D}_J(B_1, B_2) = 1 - \frac{\sum_{i=1}^{b} \min(n_i, m_i)}{\sum_{i=1}^{b} \max(n_i, m_i)} \tag{4.2}$$

Another way to approach this is to consider the fact that the sets and bags are internally encoded as vectors. Each index of the vector keeps information about one particular motion word. The set is a vector of ones and zeroes where one means, that the motion word of the given index is contained in the set and vice versa. The bags are similar but instead of ones and zeroes we put the multiplicity of a given motion word in the bag. With this representation we can use the *cosine similarity*. Again, since we need the distance, not similarity, we put *one minus* before the formula, where V_1, V_2 are vector representations of

sets (bags):

$$\mathcal{D}_C(V_1, V_2) = 1 - \frac{V_1 \cdot V_2}{\|V_1\| \, \|V_2\|} \tag{4.3}$$

The results of the experiment for *set-of-HMWs* and *bag-of-HMWs* for both distance measures are in the Section 4.6.

4.4 Introduction of weights – IDF

In the Section 4.3, we have considered all motion words to be of equal importance. In reality, this is not really the case. Imagine a motion word which is contained in every single episode. Such motion word will never help us differentiate one episode from another. In other words, it is too common. Also, a motion word which is contained many times in an episode should be more important than if it appeared in that episode only once. We will incorporate these ideas into the experiments.

Let us now quantify the importance parameter. In information retrieval, there are a few numerical statistics called *weighting schemes* which reflect how important a given word is in a collection of documents. In our case, the documents are the sequences and the words are the motion words. We will start with the *inverse document frequency* or IDF.

Inverse document frequency is derived from document frequency, which is the fraction of sequences in which the motion word is contained. We then take the logarithm of the inverted document frequency to get IDF:

$$IDF(w) = \log\left(\frac{|S|}{|\{s \in S \mid w \in s\}|}\right) \tag{4.4}$$

where S are the sequences and w is a motion word. Notice that the IDF weight of a motion word is independent of the sequence where the motion word is contained.

Now we can assign an IDF weight to any motion word. For our distance measure, we will use *cosine distance* and a weighted version of *Jaccard coefficient*. The cosine distance takes the vector representation of two *sets-of-HMWs*. If a motion word i is contained in the sequence then the vector contains the IDF weight of that word on index i otherwise

it contains 0 on this index. The weighted Jaccard distance for the *sets-of-HMWs* S_1, S_2 is a rather intuitive extension of the Jaccard distance computation from 4.1:

$$\mathcal{D}_{IDF}(S_1, S_2) = 1 - \frac{\sum\limits_{w \in S_1 \cap S_2} IDF(w)}{\sum\limits_{w \in S_1 \cup S_2} IDF(w)} \tag{4.5}$$

4.5 Introduction of weights – TFIDF

Now, we will also look at *term frequency* or TF and combine it with IDF. *Term frequency* is simply the multiplicity of a motion word in a given sequence. So if a motion word 35 is contained three times in a sequence, its weight for that sequence is three. The term frequency of motion words in a sequence contains the same information as the *bag-of-HMWs*. The difference is that we now treat it as the MW weights. The *term frequency – inverse document frequency* or TFIDF is a combination of the two. Denote $TF(w_s)$ to be the term frequency of the motion word w which is contained in a sequence s. Then TFIDF is defined as:

$$TFIDF(w_s) = TF(w_s) \cdot IDF(w) \tag{4.6}$$

It is important to note, that TF and, by extension, TFIDF are dependent on the sequence where the MW is contained, that is why we are using w_s instead of w. The sequence is important. Also, for this reason, we cannot directly use the weighted Jaccard distance from 4.5. We would need to modify it non-trivially. Therefore, we will focus on *cosine distance*, as it can be applied directly.

4.6 Experiment results

First, let us look at resulting precision of experiments without weights in Figure 4.1. The set representation works the best here slightly favouring Jaccard distance over cosine distance. However, we were able to get higher precision on the original data (3D skeletons).

We will now take the set representation with Jaccard, i.e. the one with the highest resulting precision from the no-weights experiments,

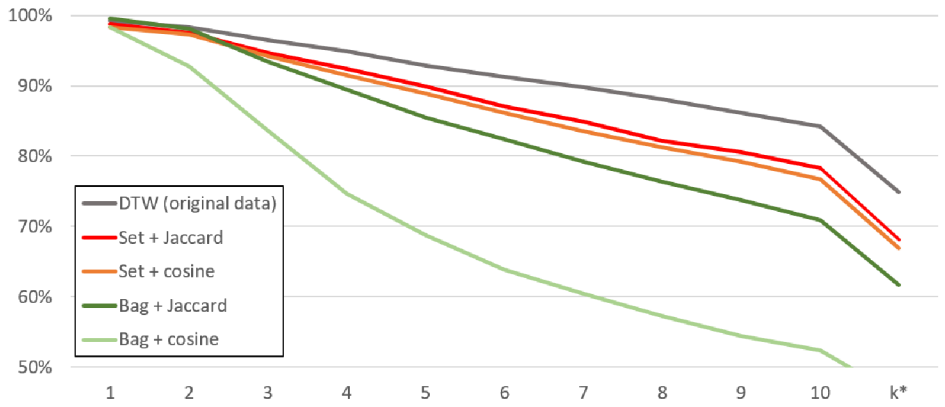


Figure 4.1: Results of experiments with HMWs without weights. The graph also contains the baseline experiment for comparison.

and compare it to the precision of the experiments which include weights. This is shown in Figure 4.2. We can see that IDF weighting gives us the best outcome beating even the DTW on the original 3D skeleton data. To be more precise, cosine distance beats weighted Jaccard distance by about one percentage point with the use of IDF. Therefore, the highest precision we can achieve with HMWs and the text retrieval techniques is using the IDF weighting with cosine distance.

Now we will put the best result into context with experiments involving DTW from 3.2. The comparison is shown in 4.3. We can see that IDF is on a par with MOMWs combined with DTW, i.e. the best result we have seen so far, and it is 30–70 times faster on our HDM05 dataset. This ratio should increase for longer episodes, since DTW is a quadratic algorithm and both weighted Jaccard and cosine distance computations are linear. Compared to the baseline experiment, we get higher precision and three orders of magnitude higher efficiency. Moreover, the HMWs and cosine distance are indexable while the MOMWs and DTW are not, which is a very important feature for practical use.

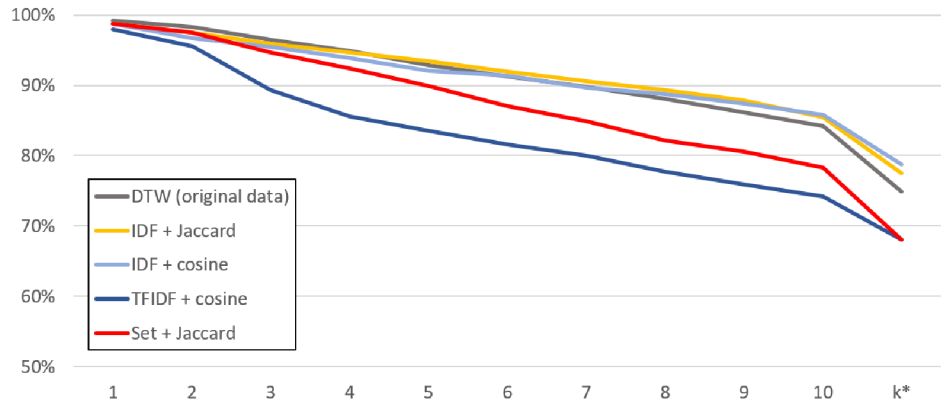
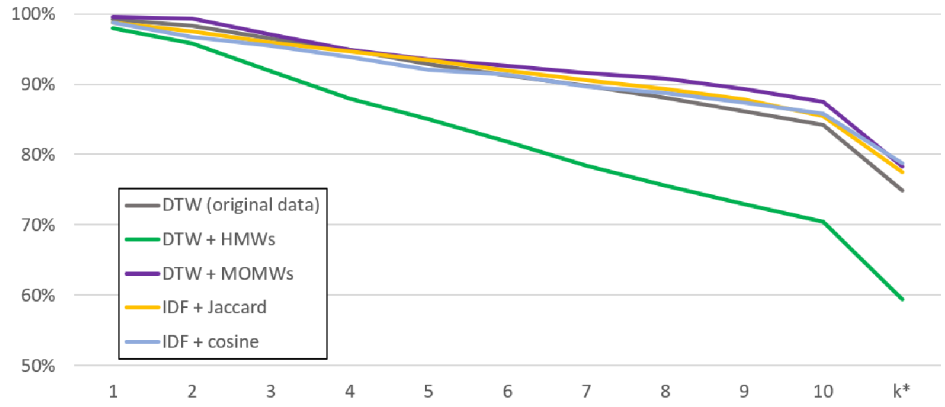


Figure 4.2: Results of experiments with HMWs with weights. The graph also contains the best result of the no-weights experiment (Set + Jaccard) as well as the baseline experiment for comparison.



	k^* precision	Efficiency
DTW + original data	74.84%	25 min
DTW + MOMWs	78.28%	37 s
DTW + HMWs	59.35%	28 s
IDF + Jaccard	77.54%	~500 ms
IDF + Cosine	78.74%	~1000 ms

Figure 4.3: Precision and efficiency comparison of the best results of text retrieval based experiments versus results of DTW based experiments.

5 Using multi-overlay motion words to improve precision

We now know that the combination of HMWs and IDF weights gives us superior results in every way compared to the baseline experiment. Let us call this experiment the *second baseline experiment*. In this chapter, we will move on from HWMs to MOMWs in an attempt to improve the precision that the second baseline experiment achieves. We have seen in 3.2 that MOMWs with DTW did better than DTW on the original data and HMWs. Also, they suffer from the border problem to a much lesser extent than HMWs. We will try similar approach as in the previous chapters. We will start without weighting the MOMWs and then try to incorporate some weights. However, there are some obstacles which we need to overcome.

The set and bag (multiset) representations, that we used in previous chapters, rely on the fact that two motion words are matching if and only if they are identical. In that case, they receive the same index in the vector representation and information about them is kept on that index. This holds for HMWs and their matching function but it does not hold for MOMWs. The MOMWs use different matching function than HMWs (see 3.2) and therefore they need not be identical to be matching. We still create a set of MOMWs for each episode in the sense that out of two identical MOMWs only one will appear in the set and out of two matching but not identical MOMWs both will appear in the set. For the representation of the MOMW set, we will use a common implementation of sets from standard libraries. We skip the cosine distance as it requires the vector representation and focus on Jaccard similarity only.

5.1 No weights

The Jaccard coefficient is also known as *Intersection over Union* (*IoU*). The intersection and union are created using the notion of identity. If a member of the first set is identical to a member of the second set then this member is a part of the union. We want to use the matching function $\mathcal{M}$ on MOMWs instead of the identity. For this reason, we

will modify the Jaccard similarity as follows. Let us denote a MOMW episode to be a set of non identical MOMWs $S = \{w_1, w_2, \ldots, w_n\}$:

$$\mathcal{D}(S_1, S_2) = 1 - \frac{\left|S_1 \overset{\triangleleft}{\cap} S_2\right| + \left|S_2 \overset{\triangleleft}{\cap} S_1\right|}{\left|S_1\right| + \left|S_2\right|} \tag{5.1}$$

where $\overset{\triangleleft}{\cap}$ is defined as

$$S_1 \overset{\triangleleft}{\cap} S_2 = \{w_1 \in S_1 | \exists w_2 \in S_2 : \mathcal{M}(w_1, w_2) = 1\}$$

5.2 Adding weights

Now we can include the weights. Let us start with IDF. In the case of HMWs, the computation of IDF was straightforward. For a given HMW, we simply needed to count in how many sequences the HMW appears, since the matching function for HMWs is a simple comparison. But for MOMWs we need to use the more complicated matching function $\mathcal{M}$ from the Section 3.2. We precompute IDF for every MOMW in every sequence individually. For the purposes of IDF computation, we say that a given MOMW w is contained in a sequence s if and only if it matches something in that sequence, i.e.:

$$\exists w' \in s : \mathcal{M}(w, w') = 1 \tag{5.2}$$

The distance function that we will use is an extension of the formula from 5.1 with added weights. Let $IDF(w)$ be the IDF weight of MOMW w:

$$\mathcal{D}_{IDF}(S_1, S_2) = 1 - \frac{\sum\limits_{w \in S_1 \overset{\triangleleft}{\cap} S_2} IDF(w) + \sum\limits_{w \in S_2 \overset{\triangleleft}{\cap} S_1} IDF(w)}{\sum\limits_{w \in S_1} IDF(w) + \sum\limits_{w \in S_2} IDF(w)} \tag{5.3}$$

where $\overset{\triangleleft}{\cap}$ is defined as before.

For the TF and TFIDF weighting schemes, we will stick to the MOMW sequence representation of the episodes, which are essentially bags (multisets) with ordering. We will ignore the ordering and use the double bracket notation, commonly used for multisets, to denote a

MOMW bag $B = \{\{w_1, w_2, \ldots, w_n\}\}$. The similarity functions for TF and TFIDF are basically the same as 5.1 and 5.3 respectively. There are just technical differences. The inputs are two bags B_1 and B_2 instead of sets S_1 and S_2, the cardinality is now defined for multisets and the one-way intersection $\overset{\triangleleft}{\cap}$ is also defined using the multiset notation:

$$B_1 \overset{\triangleleft}{\cap} B_2 = \{\{w_1 \in B_1 | \exists w_2 \in B_2 : \mathcal{M}(w_1, w_2) = 1\}\} \tag{5.4}$$

It is important to keep in mind that the bag can contain multiple copies of identical MOMWs. We can see that if a MOMW $w \in B_1$ has a match in B_2 then it is part of $B_1 \overset{\triangleleft}{\cap} B_2$ and therefore any MOMWs identical to w are also part of $B_1 \overset{\triangleleft}{\cap} B_2$. These are then added to the intersection TF times either with the weight 1, in the case of TF, or with the IDF weight, in the case of TFIDF.

5.3 Experiment results

Overall we get notable improvement in precision at the cost of efficiency compared to the second baseline experiment. The IDF weighting scheme gives us the biggest improvement with the k^* precision increasing by 10 percentage points. However, the efficiency is reduced more than 40 times. The efficiency is lower even compared to the dynamic time warping with MOMWs. We can see the results in Figure 5.1.

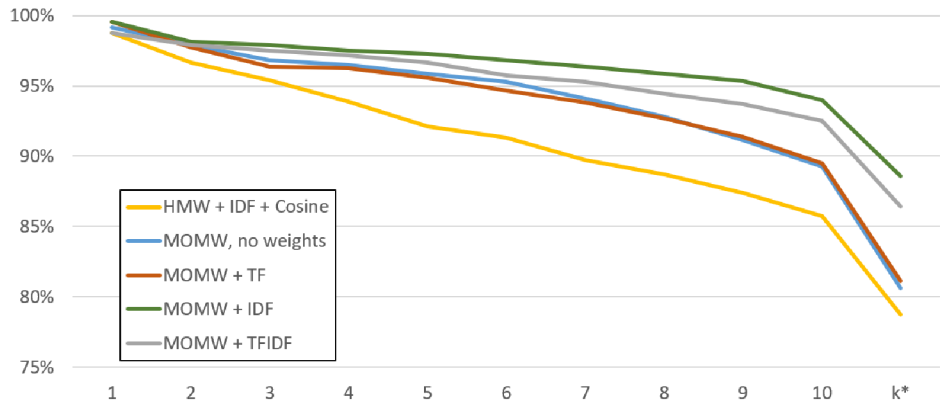
Let us look at the theoretical complexity which explains the decrease in efficiency. Let $l_{\max}$ be the size of the longest episode expressed in the number of motion words and n be the number of episodes.

- The cosine distance and Jaccard computations for HMWs are both linear, i.e. in $\mathcal{O}(l_{\max})$. We need to calculate the distance for $\binom{n}{2}$ pairs where $\binom{n}{2} \in \mathcal{O}(n^2)$. This gives us $\mathcal{O}(n^2 \cdot l_{\max})$. We still need to look at the complexity of determining IDF. To compute IDF of all present motion words, we need to go though every single motion word in every episode once. So the complexity is $\mathcal{O}(n \cdot l_{\max})$. Together the whole experiment runs in $\mathcal{O}(n^2 \cdot l_{\max})$.
- In order to compute 5.1 or 5.3 for a pair of sequences, in the worst case, we need to compare every motion word from one sequence

with every motion word from the other sequence. The complexity of this comparison is therefore $\mathcal{O}(l_{\max}^2)$ and the whole experiment for $\binom{n}{2}$ pairs takes $\mathcal{O}(n^2 \cdot l_{\max}^2)$. For 5.3, we also need to compute IDF of all MOMWs. To compute IDF, we take every sequence individually and every MOMW in it and compare it to every other MOMW in every other sequence. There are $\mathcal{O}(n \cdot l_{\max})$ MOMWs, so the IDF computation takes $\mathcal{O}(n^2 \cdot l_{\max}^2)$. Together, the experiment takes $\mathcal{O}(n^2 \cdot l_{\max}^2)$.

- Let us compare it to DTW. As we mentioned in Section 2, the DTW algorithm is quadratic, so the time complexity of a distance computation of two episodes takes $\mathcal{O}(l_{\max}^2)$. The complexity for $\binom{n}{2}$ pairs is $\mathcal{O}(n^2 \cdot l_{\max}^2)$.

We can see that 5.1 and 5.3 have the same complexity as DTW. In practice it takes 20–30% more time than DTW. However, we get significantly better precision. Because of the efficiency, or rather lack thereof, we cannot use solely MOMWs for episode matching. However, this result will still prove useful in the following chapters.



	k^* precision	Efficiency
HMW + IDF + Cosine	78.74%	$\sim 1000ms$
MOMW, no weights	80.61%	$39s$
MOMW + TF	81.15%	$42s$
MOMW + IDF	88.59%	$42s$
MOMW + TFIDF	86.42%	$44s$
IDF precomputation	N/A	$+11.5s$

Figure 5.1: Precision and efficiency comparison of MOMW experiments to the second baseline experiment.

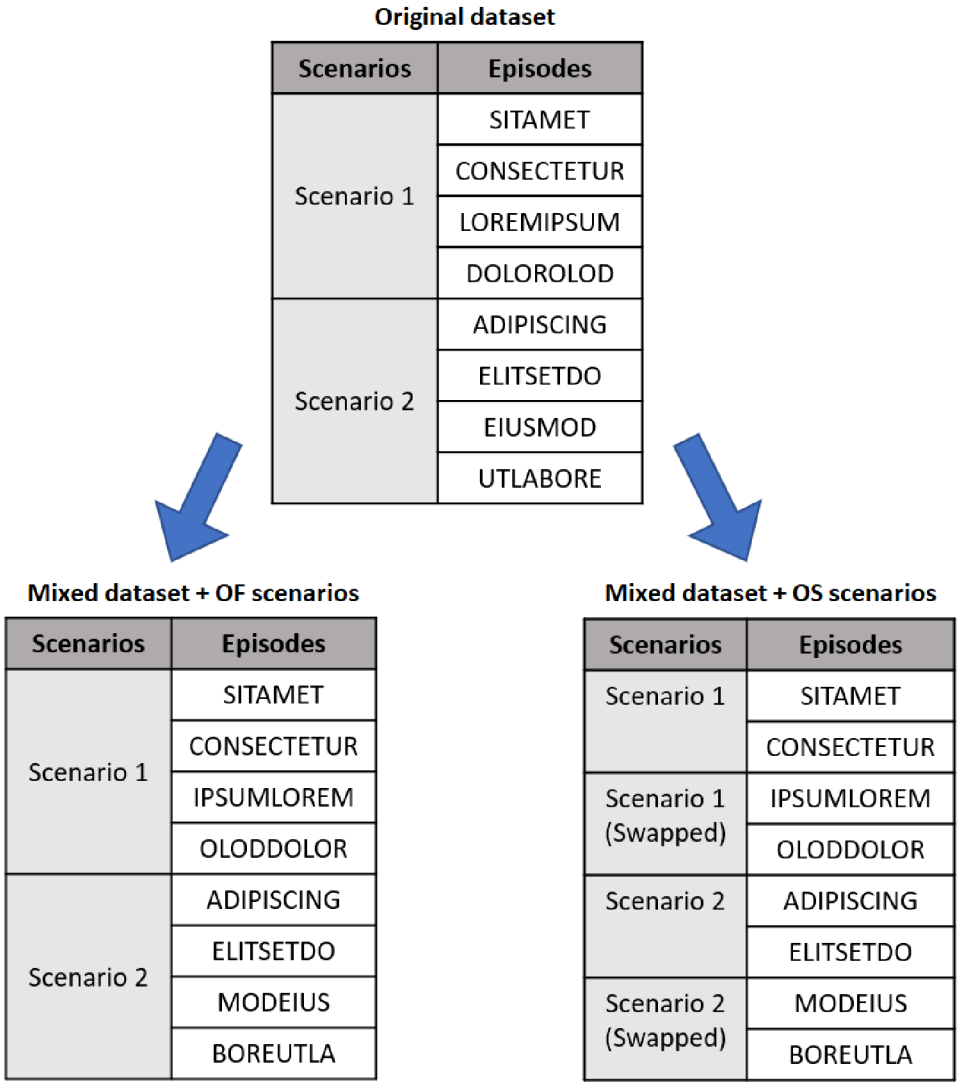
6 Considering order in episodes

Now we will look more closely at two subtypes of episode matching, the *order-free* (OF) and *order-sensitive* (OS) matching. In the previous chapters, we have used the HDM05 dataset to compare OF (bag-of-words) and OS (DTW) methods to each other. From the scenarios, we can see that any two episodes belonging to the same scenario have the same number of actions performed in the same order. This property allowed us to compare OF and OS methods to each other. However, it is reasonable to expect that not every episode matching problem will have a dataset with this property. For that reason, we will modify our current dataset, which we will now refer to as the *original dataset*, to create a new dataset, called *mixed dataset*. The mixed dataset will allow us to see the order-sensitivity of the different approaches that we have used so far.

To create the mixed dataset we take half of the episodes of every scenario and switch the first and the second half. We skip the singular episode. In other words, each scenario still has the same number of episodes but some of them are switched. If we keep the scenarios as they are, then the dataset no longer has the property, that two episodes of the same scenario contain actions in the same order. Therefore, we will call these scenarios the *OF scenarios*. We also create *OS scenarios* by remembering which episodes were switched, thus creating twice as many scenarios with halved amount of episodes per scenario. We can see this process in an example in Figure 6.1.

6.1 Episode matching using OF scenarios

We are using the same experiment from 1.5 but now with the mixed dataset with OF scenarios. We want to compare DTW on HMWs to cosine distance with IDF weighting and DTW on MOMWs to distance measure from 5.3. In the Figure 6.2 we can see that with increasing k the DTW algorithm struggles to keep up, in terms of accuracy, with the other two distance measures. It makes sense that DTW can still find mostly correct kNN for small k. However, the larger the k is the more desired nearest neighbours have switched halves which the DTW cannot identify. On the other hand, the two bag-of-MWs approaches



Original dataset

Scenarios	Episodes
Scenario 1	SITAMET
	CONSECTETUR
	LOREMIPSUM
	DOLOROLOD
Scenario 2	ADIPISCING
	ELITSETDO
	EIUSMOD
	UTLABORE

Mixed dataset + OF scenarios

Scenarios	Episodes
Scenario 1	SITAMET
	CONSECTETUR
	IPSUMLOREM
	OLODDOLOR
Scenario 2	ADIPISCING
	ELITSETDO
	MODEIUS
	BOREUTLA

Mixed dataset + OS scenarios

Scenarios	Episodes
Scenario 1	SITAMET
	CONSECTETUR
Scenario 1 (Swapped)	IPSUMLOREM
	OLODDOLOR
Scenario 2	ADIPISCING
	ELITSETDO
Scenario 2 (Swapped)	MODEIUS
	BOREUTLA

Figure 6.1: Process of modification of the original dataset (top) to create a mixed dataset with OF (bottom left) and OS (bottom right) scenarios. We can see that the number of OS scenarios doubles while the amount of episodes per scenario is halved.

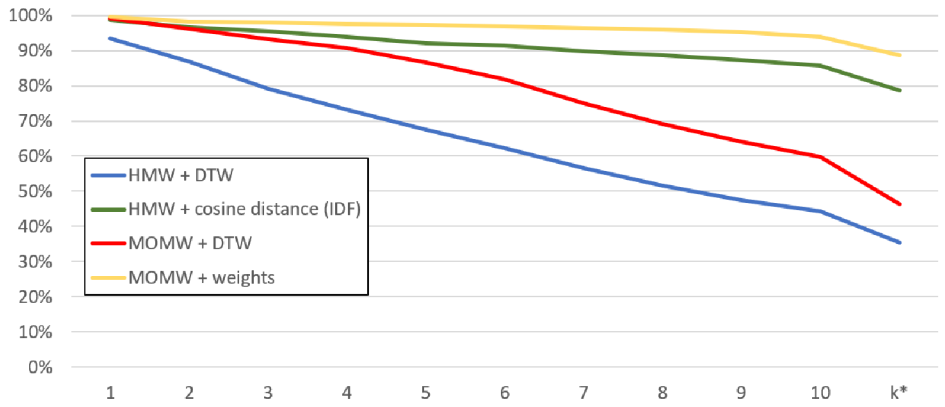


Figure 6.2: A graph showing the difference in precision of DTW for order-free episode matching compared to cosine distance for HMWs and Jaccard inspired distance measure for MOMWs.

have achieved the same results as before and swapping halves for some episodes have made no difference. In the Figure 6.3, we can see the impact the modified dataset has on DTW compared to the original dataset.

6.2 Episode matching using OS scenarios

Now we use the mixed dataset with OS scenarios. We run the same experiment but only for $k = \{1, 2, 3, 4, 5, k^*\}$, since we have fewer episodes per scenario. The value of k^* is also halved. The results of the experiment are displayed in 6.4. For DTW, we get the same gradual decrease in precision, as the k increases, that we have seen in the previous chapters. However, the other two distance measures never exceed 50% precision. That is because they cannot distinguish between episodes based on swapped and non-swapped scenarios. Therefore, on average, of all the episodes that they correctly assigned to the kNN according to the original dataset only half is still valid in the mixed dataset.

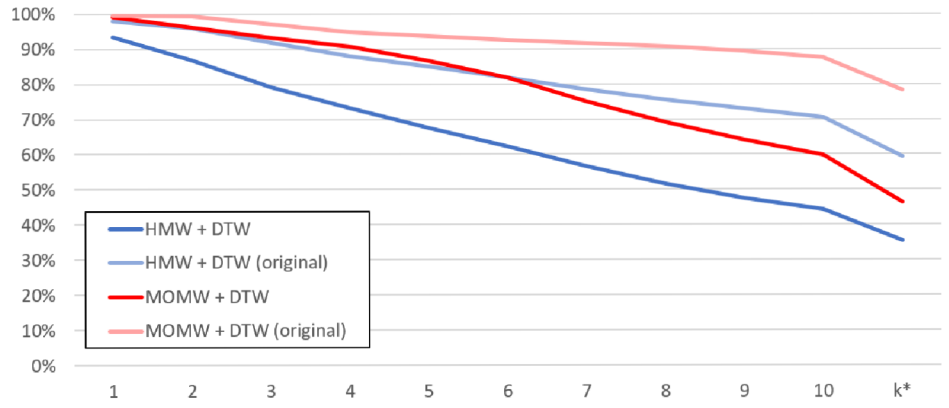


Figure 6.3: A graph showing the decrease in precision of DTW when we use the mixed dataset with order-free scenarios compared to the original dataset.

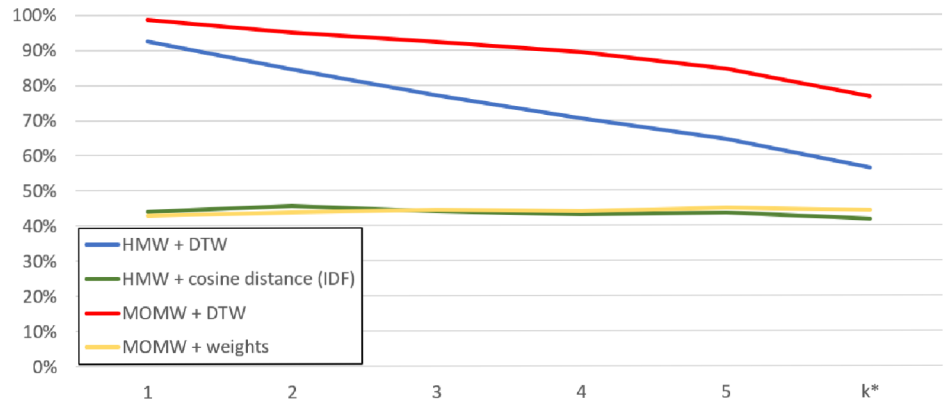


Figure 6.4: A graph showing the advantage of the sequence alignment method (DTW) for the order-sensitive matching compared to the bag-of-words approach.

7 Two-phase episode matching

We have seen, that the MOMWs are superior to HMWs in terms of precision but they need more processing power. In this chapter, we will present a *two-phase model* which exploits the indexability of HMWs and provides a very good balance between the efficiency of HMWs and the precision of MOMWs. We will primarily focus on the order-free and order-sensitive matching but at the end of the chapter, I will also comment on how the two-phase model would be used for the original dataset. In the experiment described in the Section 1.5, we have picked kNN where the k was either variable k^* or a fixed number from 1 to 10. The idea behind combining the experiments is to filter out more than kNN using HMWs and then rerank them using MOMWs and pick the kNN.

7.1 Filtering and refinement

The input for the two-phase model are episodes represented by HMWs as well as MOMWs. The output are the kNN. The two phases of the model are called *filtering* and *refinement*.

In the *filtering* phase we pick *filtering factor* denoted by f such that $f > 1.0$ and use HMWs to filter $\lceil f \cdot k \rceil$ nearest neighbours for each episode. It does not matter whether k is variable or not but we will present the results for $k = k^*$ only.

In the *refinement* phase we rerank the nearest neighbours from the previous phase using the MOMWs, meaning that in this phase, only $\lceil f \cdot k \rceil$ nearest neighbours are checked for each episode. Thanks to the parameter f we can control this number and therefore, the computation time. The output is the kNN for each episode.

7.2 Order-free matching

In Chapter 6 we have seen that the bag-of-words approach was more suitable for order-free matching. The filtering phase will use the cosine distance in combination with HMWs with IDF weighting. This makes

the filtering phase indexable and fast. For the refinement phase, we will use the weighted MOMWs with the distance function from 5.3.

We run the experiments for various f and for $k = k^*$. The weighted MOMW experiment, on which the refinement phase is based, takes around 56 seconds (see Figure 5.1). Around 25% of that time (approx. $14s$) is taken by data loading, cosine distance computation on bags-of-HMWs for filtering, but most of it is the precomputation of IDF for MOMWs. This part (denoted as *Setup* in the figure) is required and cannot be shortened. The remaining 75% (approx. $42s$) is needed for the similarity computation between all pairs of episodes (see equation 5.3). We can reduce the amount of similarity computations required by setting lower f.

We can see the results in Figure 7.1. For $f = 3.5$ the precision is 87.92%, which is less than one percentage point lower than the MOMW approach on its own. For higher f the precision grows very slowly. The parameter f has helped to reduce the number of episode pairs for which we had to compute MOMW similarity by around 82%. The whole experiment takes about 22 seconds which is more than 60% reduction compared to the pure MOMW approach, which takes 56 seconds.

7.3 Order-sensitive matching

We have shown in Chapter 6 that we need the DTW algorithm to effectively solve the order-sensitive matching. Analogically to the order-free matching we could use DTW with HMWs for the filtering phase and DTW with MOMWs for the refinement phase. However, the DTW in combination with HMWs is rather slow and does not achieve good results as is shown in 3.2. In the filtering phase, the efficiency is more important. For this reason, we do not change the filtering phase from order-free matching. The cosine distance with HMWs is more than capable of filtering out the correct episodes as it suffers only from selecting many false positives which is not an issue for sufficiently large filtering factor. The refinement phase, however, needs the DTW with MOMWs.

Our best result for this dataset so far has been achieved using the DTW with the MOMW representation, which is the red line in Figure

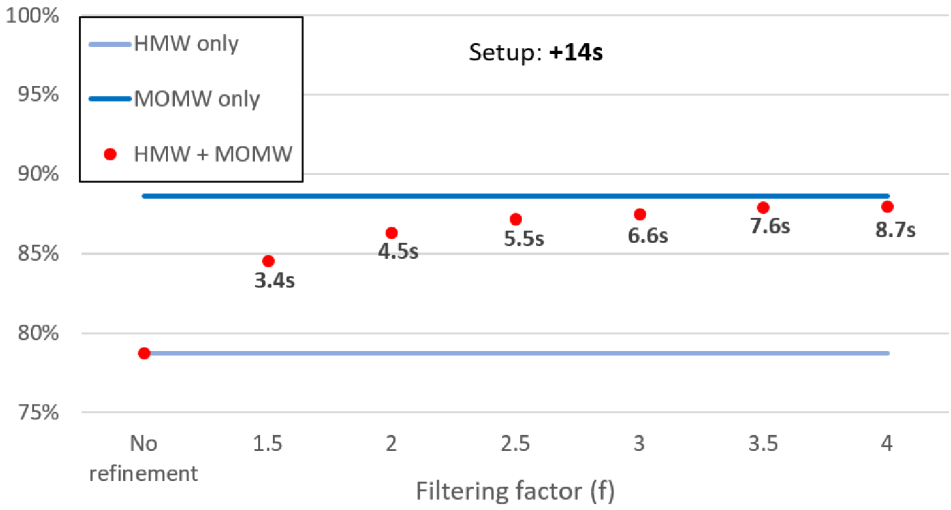


Figure 7.1: A graph showing the gradual improvement in precision of the refinement phase on prefiltered episodes for increasing value f on the mixed dataset with OF scenarios. The graph also shows the time needed to compute the similarity between the pairs of prefiltered episodes and time needed to precompute IDF for MOMWs (the setup), which is around 14s. The bottom line represents precision achieved by HMW approach, on which the filtering phase is based, and the top line represents the MOMW approach, on which the refinement phase is based.

6.4. For reference, the precision was 76.57% and it took around 36s for $k = k^*$. The goal here is to observe the effect of the filtering phase. We do the same experiment as we did for the original dataset meaning we set f to various values and k to k^*. The results are presented in Figure 7.2. We do not need any extensive setup except for the data loading and computing the cosine distances used in filtering. This takes around 1.4s. For $f = 4.0$ we achieve precision of 76.45% which is basically on par with the best result but with significantly reduced computation time at 6.1s, i.e. more than 80% reduction.

Another notable thing about the result is the sharp increase in precision at $f = 2.0$. The cosine distance used in filtering is an order-free distance measure which on average selects one false positive from the switched scenario for each true positive from the correct scenario. For $f < 2.0$ these false positives are pushing out some of the true positives from the filtered set, which then the refinement phase cannot find. At roughly $f \geq 2.0$, all of the false positives and negatives fit into the filtered set and the refinement phase can work properly.

7.4 Matching on the original dataset

We can use order-free and order-sensitive methods for the original dataset, so we simply pick the best one. For filtering, the cosine distance with HMWs and IDF still make the most sense as we used them in both cases. For the refinement we will use the weighed MOMWs rather than MOMWs with DTW since they achieved better precision. This means that the experiment is the same as in the case of order-free matching. As the difference between the original dataset and mixed dataset with OF scenarios does not influence the results of the weighed order-free approach, we get exactly the same outcome as in the order-free matching Section (7.2), which is presented in Figure 7.1.

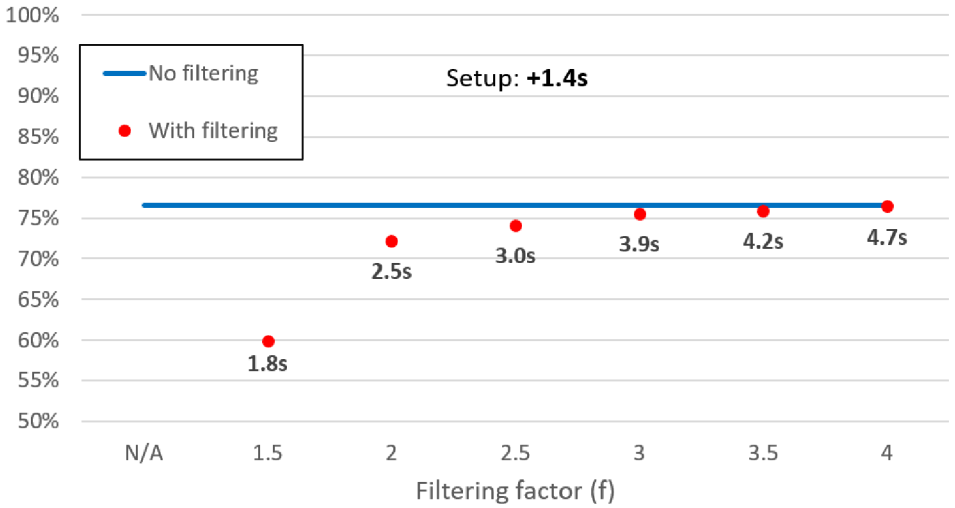


Figure 7.2: A graph showing the gradual improvement in precision of the refinement phase on prefiltered episodes for increasing value f on the order-sensitive dataset. The format is the same as in Figure 7.1, but the bottom line and the *No refinement* data point are omitted since in this case the filtering phase itself is not effective for order-sensitive matching.

8 Shingling

In the previous chapters, we have considered a single motion word to be the smallest unit. We have been creating sets of individual motion words and comparing individual motion words. However, looking at not only the motion word itself, but also on the context in which it is placed, could give us more useful information. This is commonly done by a well-known technique called *shingling* [26]. In this chapter, we will use shingling for our sequences of HMWs to factor in the surrounding motion words.

Shingling is a text processing technique which groups a certain number of consecutive letters, characters or words creating a *shingle*. The number of elements (k), grouped into a shingle, is fixed beforehand and the resulting shingles are called *k-shingles*. Shingles usually overlap, which is similar to shingles on a roof, hence the name. An example of text shingling with k set to 3 can be seen in Figure 8.1.

8.1 Shingling for sequences of hard motion words

We form *k-shingles* by taking k consecutive motion words as they appear in a sequence. We create a set of shingles for each sequence. The shingles can be *overlapping* or *non-overlapping*. The difference between them is shown in Figure 8.2. We can see that in general, we create fewer non-overlapping shingles from each sequence. The number of non-overlapping shingles goes further down, compared to the overlapping shingles, the longer the shingles are.

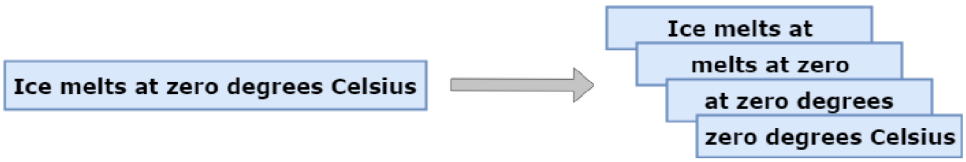


Figure 8.1: Examples of the creation of shingles from a sentence.

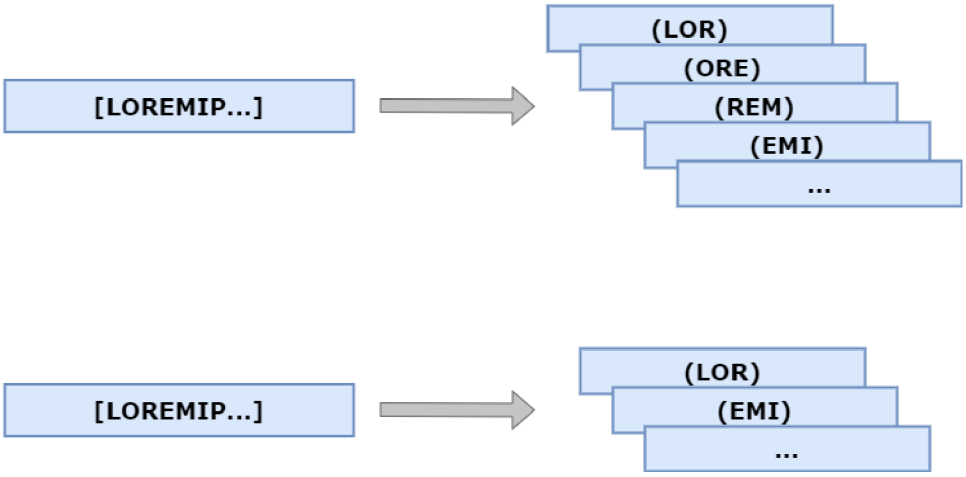


Figure 8.2: The difference between overlapping (top) and non-overlapping (bottom) hard motion word shingles.

8.2 Experiment results

Our experiments are similar to those from Chapter 4. The difference is simply that instead of sets of HMWs we will work with sets of k-shingles. Two k-shingles are considered matching if and only if the HMWs are matching on each of the k positions. We have also included the TF, IDF and TFIDF weights for the shingles. We have experimented with $k = \{2, 3, 4, 5, 6\}$ and in all cases IDF was the most successful, just like for individual HMWs. However, the precision of each of those experiments was lower than when no shingling was used (i.e. $k = 1$) for both, the overlapping and non-overlapping shingles. The resulting precision of the experiments for the IDF weighting with overlapping shingles can be seen in Figure 8.3.

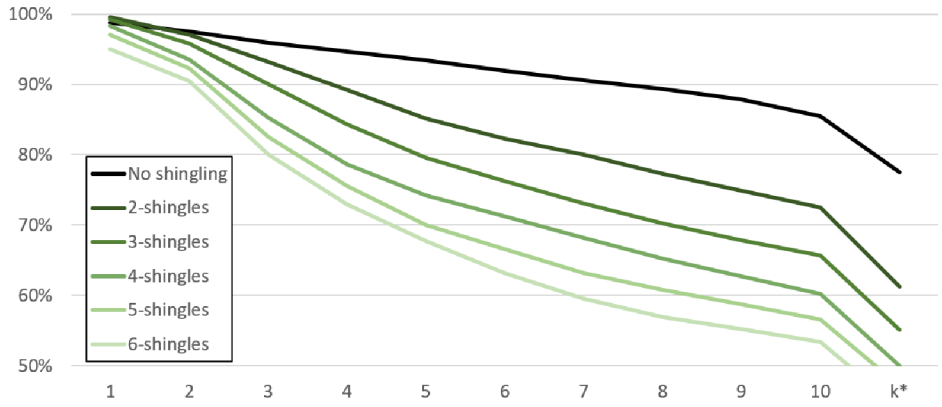


Figure 8.3: Precision of shingling experiments with overlapping shingles for 2,3,4,5 and 6-shingles with IDF weighting compared to precision of HMW expriment with IDF weighting.

9 Conclusion

We have introduced a new motion processing task called *episode matching* which deals with determining similarity between medium-length motion sequences. We have experimentally evaluated a naive (baseline) approach for this task and then improved on the results thanks to several ideas. Firstly, we have transformed the data into two more compact representations [25], the HMWs and MOMWs, to reduce dimensionality and noise which has lead to more efficient computation. Secondly, we have taken advantage of the text-like properties of the representations and have utilized several text-retrieval techniques, which have increased the efficiency and precision even further.

The main contribution of this thesis is the two-phase model, which takes advantage of higher efficiency of the HMWs and higher precision of the MOMWs. This approach has notably higher precision over the naive approach with precision increasing from around 75% to almost 88%. The computation time has improved by more than two orders of magnitude compared to the naive approach. For order-sensitive episode matching we can equip the second phase of the model with a sequence alignment function such as DTW. Another important aspect of the two-phase model is that it does not require any labelled data. We have used the scenarios from HDM05 dataset as labels but only to evaluate the effectiveness and efficiency of our experiments, not for the construction of the models themselves.

The two-phased model would benefit greatly from an efficient indexing structure, which is something that we have not touched on. The first phase of the model uses HMWs, which are scalars, and therefore allow for an efficient indexing. However, the MOMWs and the DTW algorithm are not trivially indexable which hampers the performance. More research dedicated to determining viable indexing techniques would greatly support the efficiency of the episode matching task as well as the scalability of our solution. Besides indexing, bigger and higher quality datasets would also help us gain better insight into the problem. We have considered artificially enlarging the dataset using augmentation techniques from [27]. The authors propose four: crop, extension, shift and adding noise. However, these techniques are meant for actions contained in longer (so-called parent) sequences.

The extension and shift operations assume the existence of the parent sequence, which is not present in the case of episodes. The crop operation creates strict subsets in the bag-of-words approach, which is not desirable and the noise addition rarely changes motion word assignment during clustering. However, a promising way of obtaining a good quality datasets seems to be the estimation and extraction of human poses from ordinary videos.

Bibliography

1. GRAY, A. A Brief History of Motion-Capture in the Movies. *IGN* [online]. 2014 [visited on 2020-11-17]. Available from: `https://www.ign.com/articles/2014/07/11/a-brief-history-of-motion-capture-in-the-movies`.
2. ALLDIECK, T.; MAGNOR, M.; BHATNAGAR, B. L.; THEOBALT, C.; PONS-MOLL, G. Learning to Reconstruct People in Clothing From a Single RGB Camera. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 1175–1186. Available from DOI: `10.1109/CVPR.2019.00127`.
3. CAO, Z.; SIMON, T.; WEI, S.; SHEIKH, Y. Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 1302–1310. Available from DOI: `10.1109/CVPR.2017.143`.
4. SUN, K.; XIAO, B.; LIU, D.; WANG, J. Deep High-Resolution Representation Learning for Human Pose Estimation. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 5686–5696. Available from DOI: `10.1109/CVPR.2019.00584`.
5. KREISS, S.; BERTONI, L.; ALAHI, A. PifPaf: Composite Fields for Human Pose Estimation. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 11969–11978. Available from DOI: `10.1109/CVPR.2019.01225`.
6. AHMAD, Z.; KHAN, N. Towards Improved Human Action Recognition Using Convolutional Neural Networks and Multimodal Fusion of Depth and Inertial Sensor Data. In: *2018 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2018, pp. 223–230. Available from DOI: `10.1109/ISM.2018.000-2`.
7. KE, Q.; BENNAMOUN, M.; RAHMANI, H.; AN, S.; SOHEL, F.; BOUSSAID, F. Learning Latent Global Network for Skeleton-Based Action Prediction. *IEEE Transactions on Image Processing*. 2019, vol. 29, pp. 959–970. Available from DOI: `10.1109/TIP.2019.2937757`.

8. SONG, S.; LAN, C.; XING, J.; ZENG, W.; LIU, J. Spatio-Temporal Attention-Based LSTM Networks for 3D Action Recognition and Detection. *IEEE Transactions on Image Processing*. 2018, vol. 27, no. 7, pp. 3459–3471. Available from DOI: 10.1109/TIP.2018.2818328.

9. BUDIKOVA, P.; SEDMIDUBSKY, J.; HORVATH, J.; ZEZULA, P. Towards Scalable Retrieval of Human Motion Episodes. In: *22nd IEEE International Symposium on Multimedia (ISM)*. IEEE Computer Society, 2020, p. 8.

10. LARABA, S.; BRAHIMI, M.; TILMANNE, J.; DUTOIT, T. 3D skeleton-based action recognition by representing motion capture sequences as 2D-RGB images. *Computer Animation and Virtual Worlds*. 2017, vol. 28, no. 3-4, pp. e1782. Available from DOI: https://doi.org/10.1002/cav.1782. e1782 cav.1782.

11. LIU, K.; GAO, L.; KHAN, N. M.; QI, L.; GUAN, L. A Multi-Stream Graph Convolutional Networks-Hidden Conditional Random Field Model for Skeleton-Based Action Recognition. *IEEE Transactions on Multimedia*. 2020, pp. 1–1. Available from DOI: 10.1109/TMM.2020.2974323.

12. LIU, R.; XU, C.; ZHANG, T.; ZHAO, W.; CUI, Z.; YANG, J. Si-GCN: Structure-induced Graph Convolution Network for Skeleton-based Action Recognition. In: *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8. Available from DOI: 10.1109/IJCNN.2019.8851767.

13. LIU, J.; WANG, G.; DUAN, L.; ABDIYEVA, K.; KOT, A. C. Skeleton-Based Human Action Recognition With Global Context-Aware Attention LSTM Networks. *IEEE Transactions on Image Processing*. 2018, vol. 27, no. 4, pp. 1586–1599. Available from DOI: 10.1109/TIP.2017.2785279.

14. WU, Y.; WEI, L.; DUAN, Y. Deep spatiotemporal LSTM network with temporal pattern feature for 3D human action recognition. *Computational Intelligence*. 2019, vol. 35, no. 3, pp. 535–554. Available from DOI: https://doi.org/10.1111/coin.12207.

15. ELIAS, P.; SEDMIDUBSKY, J.; ZEZULA, P. A Real-Time Annotation of Motion Data Streams. In: *2017 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2017, pp. 154–161. Available from DOI: 10.1109/ISM.2017.29.

16. CARRARA, F.; ELIAS, P.; SEDMIDUBSKY, J.; ZEZULA, P. LSTM-based real-time action detection and prediction in human motion streams. *Multimedia Tools and Applications*. 2019, vol. 78, pp. 27309–27331. Available from DOI: 10.1007/s11042-019-07827-3.

17. SEDMIDUBSKY, J.; ZEZULA, P.; SVEC, J. Fast Subsequence Matching in Motion Capture Data. In: *Advances in Databases and Information Systems*. Springer International Publishing, 2017, pp. 59–72. ISBN 978-3-319-66917-5. Available from DOI: http://dx.doi.org/10.1007/978-3-319-66917-5_5.

18. SEDMIDUBSKY, J.; ELIAS, P.; ZEZULA, P. Searching for Variable-Speed Motions in Long Sequences of Motion Capture Data. *Information Systems*. 2018, vol. 80, pp. 148–158. Available from DOI: 10.1016/j.is.2018.04.002.

19. LIU, C.; HU, Y.; LI, Y.; SONG, S.; LIU, J. PKU-MMD: A Large Scale Benchmark for Continuous Multi-Modal Human Action Understanding. *CoRR*. 2017, vol. abs/1703.07475. Available from arXiv: 1703.07475.

20. SEDMIDUBSKY, J.; ELIAS, P.; ZEZULA, P. Benchmarking Search and Annotation in Continuous Human Skeleton Sequences. In: *Proceedings of the 2019 on International Conference on Multimedia Retrieval*. Ottawa ON, Canada: Association for Computing Machinery, 2019, pp. 38–42. ICMR '19. ISBN 9781450367653. Available from DOI: 10.1145/3323873.3325013.

21. MÜLLER, M.; RÖDER, T.; CLAUSEN, M.; EBERHARDT, B.; KRÜGER, B.; WEBER, A. *Documentation Mocap Database HDM05*. 2007. ISSN 1610-8892. Technical report. Universität Bonn.

22. MÜLLER, M. 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.

23. PROGRAMMINGLINGUIST. *Dynamic time warping*. 2015. Available also from: https://commons.wikimedia.org/wiki/File:Dynamic_time_warping.png. [Online; accessed 03-July-2020].

24. ARISTIDOU, A.; COHEN-OR, D.; HODGINS, J. K.; CHRYSANTHOU, Y.; SHAMIR, A. Deep Motifs and Motion Signatures. *ACM Trans. Graph.* 2018, vol. 37, no. 6, pp. 187:1–187:13. Available from DOI: 10.1145/3272127.3275038.

25. SEDMIDUBSKY, J.; BUDÍKOVÁ, P.; DOHNAL, V.; ZEZULA, P. Motion Words: A Text-Like Representation of 3D Skeleton Sequences. In: *Advances in Information Retrieval*. Springer, 2020, pp. 527–541. ISBN 978-3-030-45438-8. Available from DOI: 10.1007/978-3-030-45439-5_35.

26. LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. Shingling of Documents. In: *Mining of Massive Datasets*. 3rd. USA: Cambridge University Press, 2014, pp. 78–80. ISBN 1107077230.

27. SEDMIDUBSKY, J.; ZEZULA, P. Augmenting Spatio-Temporal Human Motion Data for Effective 3D Action Recognition. In: *21st IEEE International Symposium on Multimedia (ISM)*. IEEE Computer Society, 2019, pp. 204–207.