How To Leverage Spatial Trajectory Data To Identify Work Practices

Echbiki, Omar

Award date:
2021

Department of Mathematics and Computer Science
Process Analytics

How To Leverage Spatial Trajectory Data To Identify Work Practices

Master Thesis

Omar Echbiki

Supervisors:
Felix Mannhardt - *TU/e*
Simon J. Julier - *KIT-AR Ltd*

16-AUG-2021

Abstract

With the increment of motion detection and tracking devices used in the industry, the abundance of data that these devices create is unprecedented. Using the data provided by these devices can provide a deeper look inside the processes of manual work related tasks. Specifically, motion data, that is information about the location of a person in time. In this work, we use this data in the context of process mining, specifically to detect and improve the manual processes for work practices. In these manual processes, workers execute a given task or activity that are part of a broader case execution. This work tries to join process mining and sensor data to take a deeper look at what is happening within a task, apply process mining techniques to it and improve manual work processes using the motion data. To do so, we use Dynamic Time Warping on the motion data on an activity level and spatio-temporal clustering on a case level. This provides us with granularity and can be used to compare cases individually. To validate this algorithm, two datasets are used for evaluation. The first coming from a real life scenario from the company KIT-AR using AR glasses to help in the execution of manual processes of companies. We also make use of the Karate dataset composed by a set of Karate athletes of different skills executing karate moves. The athletes have different age, height and skill level. Our approach successfully clusters the data based on the motion and the other metrics such as time, using agglomerative clustering with precomputed pairwise distances. Silhouette method is used for choosing the weights and the number of clusters. Our implementation was efficient in finding anomalies in the execution of the tasks as well as in detecting different work practices and identified problems in the data collection. Finally, when we compare the process graph of each cluster, we see different ways in the working practice and the activities, finding anomalies in the data as well. This approach offers a first look in using 3D motion data to identify work practices, previous literature in the field either did not use motion data or used it for activity recognition.

Contents

Chapter 1

Introduction

In recent years, an increasing number of wearables, in particular Augmented Reality glasses (AR glasses), are being widely used in industrial setting to aid workers. One of the uses for this technology is to help workers in their manual work with instructions, indications and by showing them virtual objects.[1] These wearables produce an unprecedented amount of data, coming from different kinds of sensors, in particular of position tracking. When exploiting this data, we can extract the location and the trajectory of users wearing them, and it allows us to compare these trajectories across multiple users and across different executions of the same task in different time. Giving us a detailed look at the execution of tasks, information like duration of the task and movement being done during this execution. Particularly interesting is to compare them with each other to find differences in the process itself. The advantage of using 3D motion data is that you can represent motion using points in space and only preserve the movement, without having to record the entire video of the activity. This data has been very useful and has application in various fields, ranging from computer vision to process film-making or video games, healthcare, social sciences and in industry which is the fields we focus on [2] [3].

Specifically 3D motion data coming from AR devices in factory environment, where these repetitive task are more likely to appear and be labelled, and the task can happen again very similarly. The research question that this work tries to solve is *how can we leverage 3D motion into identifying different work practices and view the data with a process lens.*

The work is divided into two steps, the first focusing on the tasks. A task is a basic activity being executed by the actor or resource, an example of tasks in the context of an industrial setting can be: open lid, screw panel, execute diagnostic and many more. The goal is to compare task executions and efficiencies with all the other tasks. This is done using Dynamic Time

Warping, a method for comparing time series that can adapted for motion trajectories and align them even if the two trajectories have different lengths and speeds. This is especially a problem when considering human motion data that can vary a lot from individual to individual. This method is used to find the distance between those two series [4]. To validate the approach, we use both the KIT-AR dataset, introduced in section 3.2, coming from real-life work environment and synthetic data from Karate techniques dataset [5]. The reason we make use of a second dataset to check the validity is because we do not have a ground truth in the case of the company data, therefore it is hard to determine if the comparison is correct or not.

The second part will focus on the general case execution. A case is a sequence of ordered events or activities. The order of activities across cases will tell us how different is each case from the others.

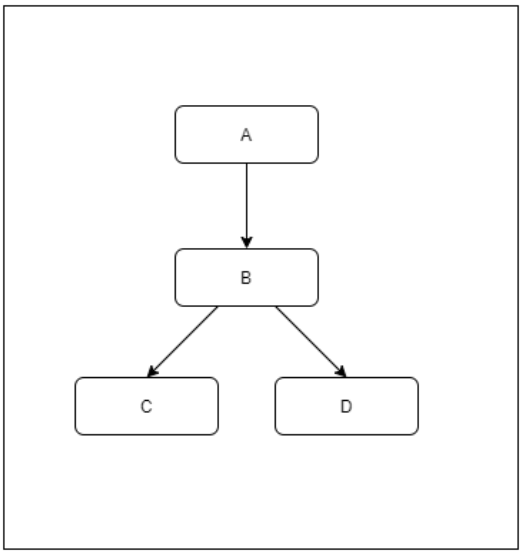
In this part, we cluster them based on similarity parameters, one of which is the motion similarity computed in the first part. We make use of other features such as execution time and process information. The goal of this phase is to take into account different metrics to calculate the similarity matrix and ultimately identify different work practices. The clustering uses different attributes that depend on the context and the industry. We use a modified formula from [6] that, instead of using the Euclidean distance, uses the Dynamic Time Warping and other distance measures. After calculating the pairwise distances between all the cases, we can assign weights for each metric and find the number of clusters. To determine these values, we use the Silhouette method [7] with different values for each parameter, and we choose the parameters that have the highest silhouette score, meaning that the clustering is more successful.

1.1 Context and Topic

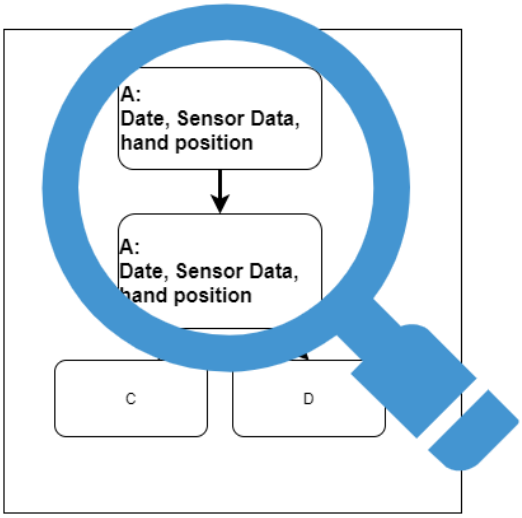
In traditional Process Mining, it is common to look into the process as a whole, meaning that the only available information is about the activities within the case. The basic component to do any process mining related analysis relies on *event logs*. These are a series of tasks executions that are part of a case. Each *task* (or *event*) refers to a *case* execution. Usually each event in an event log is characterized by an activity name, timestamp, the resource carrying the task, whether it's a human or a machine, but there is no further information on how the task is executed. Being able to understand how the task is executed could provide us with greater insight on the work practices, the efficiency and the bottlenecks within a task execution that can arise. Providing us with unprecedented granularity over the process as a

whole.
The goal of this thesis is to address the limitation of traditional process mining information and make use of the data coming from sensors and devices that can be very useful, specifically we focus on the motion data coming from Microsoft HoloLens [1].

[1] ¡https://www.microsoft.com/en-us/hololens/hardware¿



(a) Traditional process mining



(b) process mining augmented with sensor data

Figure 1.1: How our approach would give us more detail inside the process visualized using Directly Follows Graph formalism

The image 1.1 shows a very simplified logic on how the approach needed is helpful for gaining a better understanding of the process, mainly with motion data. In traditional process mining the information available outside the task, case and resource is limited. However, being able to "look into" the execution of the activity can provide answers to questions about the task. Before this data was available, we were only able to answer which activity

being executed, the date and in some cases, the resource executing the task. Now in addition to all the information we had before we can also find out the way the task is being executed to the point of knowing the exact movement of the actor for the whole duration of the task. Having this much data over the process offers a lot of potential in enriching the process information and the ability to find problems and optimize the tasks [8].

1.2 State of the Art

As we mentioned before and as far as we know, there are no methods currently that use motion data and integrate it with process mining. However, in [9] the authors proposed a method to use sensor data to improve on the product design process by transforming the sensor data into event logs. The approach used there was to segment a continuous stream of data coming from the sensors and use it to create the event logs to understand how the product was being used and how they can improve on the product design. The main difference between the approach we are using is that we already have a clear separation of tasks, and we use the sensor data to look into the event execution through motion data coming from the HoloLens sensors.

In the field of manual assembly and process mining, in [8] the authors proposed a tool that uses sensors and images from a camera and uses the data collected to train an AI algorithm to automatically detect movements like picking and placing an object. In comparison to our approach, this is only validated on a set-up environment and not in real-life scenario. Another difference is that they use their approach for activity recognition and do not compare them to each other and detect different work practices.

There is therefore the need to use motion data to augment and use with process mining to allow for more fine-grained analysis, the main focus is to compare different 3D motion data with each other and ultimately being able to identify different work practices within the process logs.

A complete overview of the state-of-art and related work is provided in section 2.2.

1.3 Research Question

Based on the state of the art and current technologies, our approach will focus on how to compare different executions of a certain task, by only having the motion data information from the sensors. As mentioned before, we will be using Microsoft HoloLens 2 and this headset provides us with point location

across time for each joint of the hands (when visible by the RGB-D camera) and for the headset itself. In Figure 1.2 we can see where the depth sensor is located, this is important to understand that the hand's position will only be visible when the hands are in sight. This particular model is used in the context of KIT-AR, a company that provides AR solutions in industry. The data collected from one of their projects is the main dataset that we focus on to answer our research question.

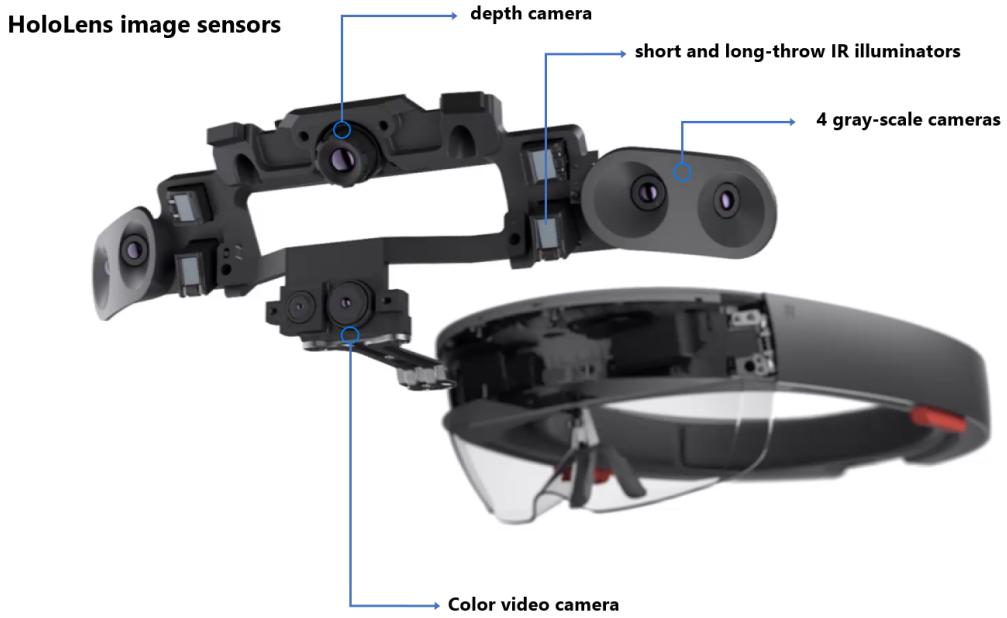


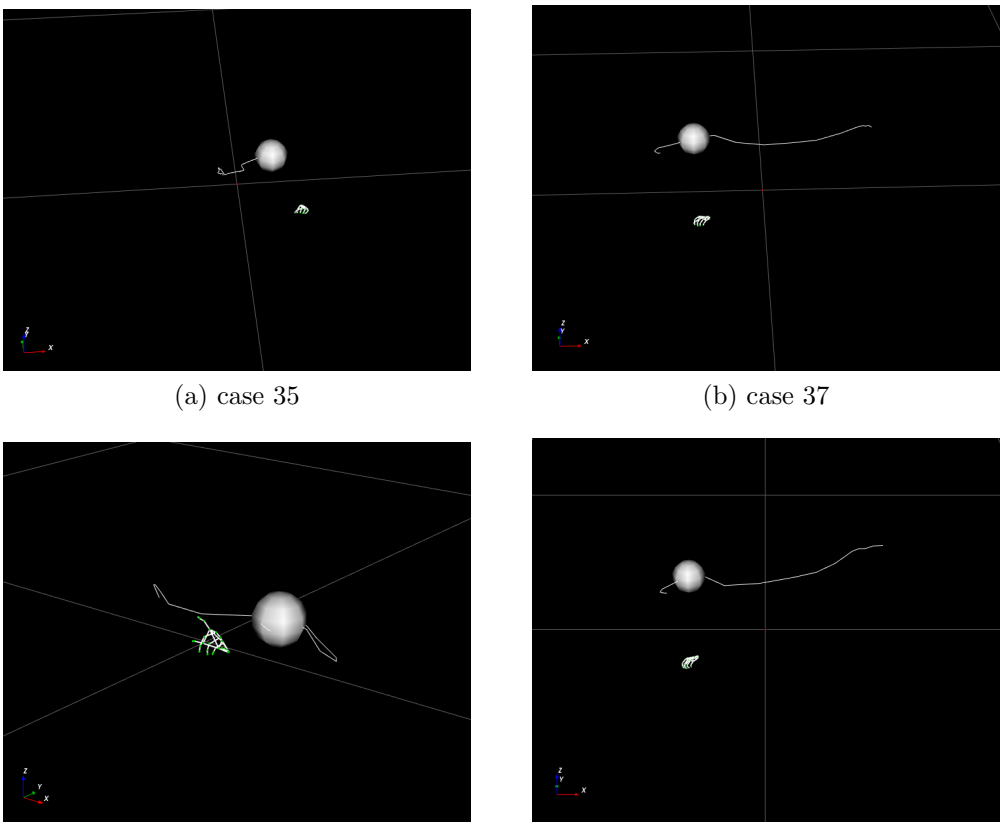
Figure 1.2: Microsoft HoloLens 2 - Overview of sensor and components *(Image from wevolver.com, 2021)*

For this given dataset coming from a real life scenario, we will use the motion data and other information to detect different work practices. Since this dataset won't have a ground truth, it won't be possible to evaluate the results. That is why in addition to the KIT-AR dataset we make use of a Karate dataset, that has some more information that we can use to evaluate the success of our approach, at least in the first phase. The research problem will be divided into two sub problems:

The first one being how to compare two different 3D human motion data, or trajectories. This problem requires an approach able to find similarities of work done by different people, at different speeds and with different heights. Another requirement for this first phase is that the algorithm needs to work with different lengths of sequences, since it is coming from real-life data, it is

rarely the case that two sequences have the same lengths. Additionally, we need other metrics to assess difference between cases.

The second problem is, once we have the difference in human motion data and eventually other metrics, to cluster and retrieve work practices. By work practices, we mean different ways of executing a certain task or a set of tasks, specifically from a human motion point of view, that we will define as a case. For example, for executing the task "LiftCovers" found in the KIT-AR dataset there might be different paths takes to do the same thing, as shown in figure 1.3. Some are very similar to each other, like **b** and **d**, and some are very different. Proving that one task can have different ways of being executed.



(a) case 35 (b) case 37

(c) case 40 (d) case 57

Figure 1.3: Different movements to execute task **LiftCovers** - *data extracted from KIT-AR system and visualized using Mokka tool*

1.4 Method or Approach

In this section, we introduce the method used and the approach taken to tackle the research question.

For the first part, we focus on exploring the dataset and understanding it, plotting it using tools like Plotly [10] on a Jupyter Notebook running on a remote machine where the dataset is stored. For this part, we use, in addition to the KIT-AR data, the karate dataset. In order to find different work executions inside the dataset, we need to have first a way of comparing two instances of a work execution, that we call *task/activity*. This activity or task is stored in the dataset and provides us with detailed position of each of the joint and the headset position. An overview of some type of data that is present in the dataset is shown in Figure 1.4.

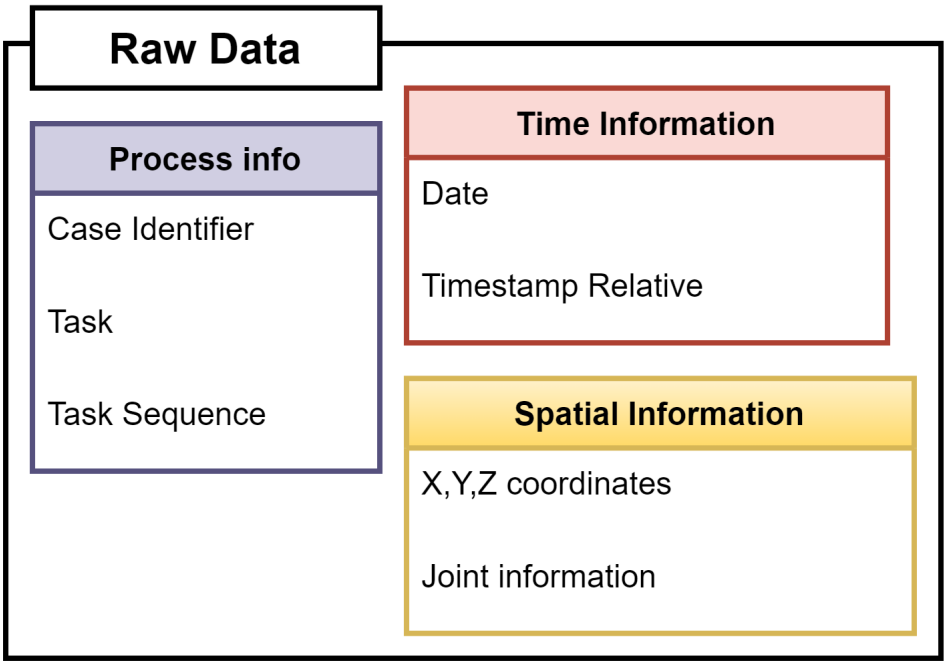


Figure 1.4: Overview of the data types in KIT-AR dataset

During the exploratory phase, we convert the data into C3D format. This format is widely used as the standard format to represent bio-mechanical motion data [11]. This is particularly useful when used in combination with Mokka tool to visualize this type of data, since it shows motion and provides a better understanding compared to plotting in a 3D plot [12]. Figure 1.3 shows an example of the visualization of the joints in a temporal manner. In

Mokka, this is done using the concept of frames, where each frame represents a position in space of the given joints at a specific time. Before being able to visualize the joints and the executions, we convert the data points into joints location using the format c3d. The next step is to give it a human-like visualization. The headset location has the bigger spherical node of a radius 30 times the size of the fingers. This is done to reflect the proportions of the average human body. Furthermore, we assign different colours to the left and right hands to make it easier to identify in a 3D environment. In order to compare and analyse these motion trajectories made by these points, we need a way of comparing 3D trajectories. To do so, we make use of Dynamic Time Warping, an algorithm that compares two sequences of points to find the best alignment and computes a distance between the two. This will be one of the metrics we use for the algorithm and the final comparison. The Dynamic Time Warping or DTW tries to find the best alignment regardless of the speed of the task execution, or physical features of the person executing the task. In addition to the DTW we will make use of another metric to compare two execution, one that will make use of the temporal dimension of the data, that will be to compute the temporal difference based on timestamps and the date, both present in the data as shown in Figure 1.4. Since it is hard to evaluate the algorithm on the Dataset of KIT-AR, to check if our algorithm is working we make use of the karate dataset [5] where we have techniques and motion that repeat similarly across observation, and it comes with detailed information on the athlete executing the tasks. The karate dataset was very helpful in the first part, since it has motion data and knowledge about the athletes. However, for the second part we needed a dataset with process information and motion data, that we could not find.

Once we defined the first two metrics that refer to the spatial and temporal dimension we need to take into account the process mining aspect of things, that means accounting for difference in the order of tasks within a case, whether a task is missing or if there is one task in excess. We make use of Levenshtein distance on the tasks of a given case, and we compare them with the task of the reference case at that given moment [13].

The second part is computing an overall distance using general distances such as the spatial and temporal distance and also attribute distances that can change based on the domain, such as the Levenshtein distance on the traces between each case.

The next step is to compute the pairwise distances between all cases in the dataset and identify the different work practices. The formula for this has weight for each of the distances, and to find these weight we use the Silhouette method that tells us what are the best values for the weights of each metric and determines the best K value for clustering.

The final step is then to compute a Directly follows graph or DFG describing the process of cluster of cases, that are ultimately the work practices i.e. how people execute these work, and we discuss the results.

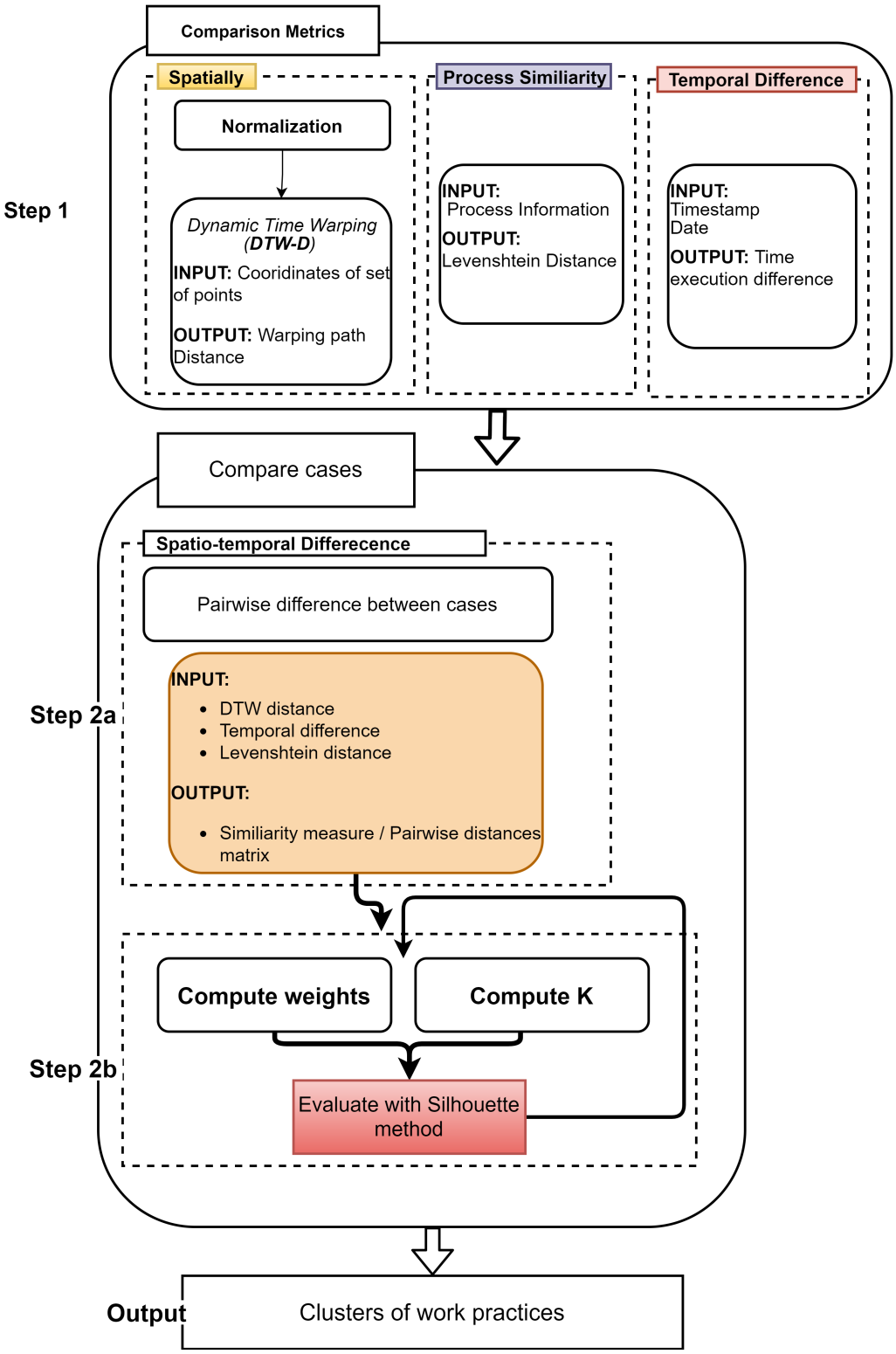


Figure 1.5: Overview of the general process and the different phases

Picture 1.5 shows an overview of the various steps of our approach using the raw data introduced in Figure 1.4.

In **step 1**, we define the three comparison metrics:

1. Spatial dimension
2. Temporal similarity
3. Process similarity

For the spatial dimension, we use Dynamic Time warping, this gives a similarity measure regarding the movement and the trajectory created by the motion in time. This outputs a distance that tells us how much the first trajectory differs from the second, the closer the trajectory the smaller the DTW distance is. This is done for each task within the case, and then aggregated to produce the difference between two cases.

For the temporal dimension, we use a simple time difference approach. For each case in the dataset we calculate the time required to execute each task by using the value of the timestamp and the date. This is then compared with the execution time of the other case we are comparing to and the difference is the result of this step.

Finally, for the process similarity we use Levenshtein distance, a measure of how the tasks of each case we are comparing differ with each other. This accounts for how different case have either a different order or are missing some tasks.

The outputs of this step becomes the input for **step 2a**. In this step, we compare the cases, joining all the similarity measure into one value. This value indicates the similarity on all three metrics of the two given cases.

However, before having the final similarity measure in **step 2b** we compute the best values for the weights for each of the three metrics in **step 1**, and we compute the best value for the number of clusters.

The output of step 2 as a whole is the similarity measure between the cases with the optimal values for the weights and the number of clusters. When applied to the whole dataset with all the cases, we have a similarity matrix.

1.5 Findings

In this section, we introduce the main results of the approach and discuss what they mean briefly.

Results

Based on the dataset we have, and after applying the steps described above we come up with clusters of work executions, however this was hard to evaluate since in the data itself we do not have any ground truth. That is why we contacted the domain expert working inside the company to assess discuss with him the finding.

The algorithm identified 3 main clusters of execution, with the weights given to each metric. The result was that the Levenshtein distance has the most weight among the three metrics.

Interpretation

The three clusters identified by the algorithm were largely due to the Levenshtein distance, since it was given the highest weight among the distances. This explains why the 3 clusters differ in the process execution or in the traces, since every case in one of the clusters has a different number of tasks, or missing tasks, compared to another cluster.

Some key findings are that the algorithm identified two clusters where the process differed in one specific task: **PhotoCapture**, when visualizing these tasks using Mokka we notice that the location of the headset, provided by the sensor, was not present in more than 91% of the cases, this contributed to create two separate clusters of executions. On the other hand, when checking the other cluster, all the activities in that cluster had correct reading of the HoloLens Headset location sensor for that specific task. In addition, the algorithm also successfully identified an outlier in one of the three clusters where an activity used in case of error was heavily present in that case.

The rest of this thesis is organized as follows: In section 2.2 we introduce the main concepts and definitions used throughout this thesis and the related work in the fields of search in motion data and the use of sensor data in process mining. In section 3 we introduce the problem more in depth with focus on the company dataset and the synthetic dataset. In section 4 we introduce DTW algorithm and some basic usage, before applying it to the real data, and in section 5.2 we use this metric and introduce the other metrics, we present the results in section 6.4 before evaluating it in section 6.5 using silhouette method. In the same section we discuss the result and show the DFG of the activities, finally we conclude with some reflections on the work in section 7.

Chapter 2

Background

In this section, we introduce the terms that will be useful and that require a definition to better follow the rest of the thesis, in the second part we review the related work.

2.1 Preliminaries

Definitions

- **Event/activity/task**: Refers to an activity that is executed. In the context of industry, for example, the events or activities can be steps in the production line. We use task/event/activity as synonyms in the rest of the thesis.
- **Event Log**: A sequence of ordered events, belonging to the same case execution.
- **Joint**: It is the smallest point that is detected with the camera motion, a set of joints form a body part, for example Hand, or finger and are set in a hierarchical way: *Index tip* < *Index* < *Left-hand*
- **Trajectory/motion data**: A trajectory is defined as a set of temporally ordered points in 3-dimentional space R^3. Given a trajectory t and a time i, the trajectory *t(i) = (x,y,z)* describes the position of the point at time i for the duration of the task.
- **Case**: Is a sequence of tasks or activities, identified with a unique id.
- **AR glasses**: *Augmented Reality glasses* like the Microsoft HoloLens. These devices are used to visualize virtual objects and integrate them with the physical environment.

2.2 Related Work

We begin by describing the three relevant lines of research correlated with our thesis:

1. Use of motion data in process mining
2. Similarity search in motion data
3. Grouping and clustering in spatial and temporal dimensions

Use of sensor data in process mining. We analyse the different uses of sensor data in the field of process mining. As mentioned before, we did not find a lot of literature studying the use of motion data, but rather a lot of the focus is on mapping sensor measurement to human. In that field, the first paper we analyse the work by [9]. The authors here use sensor data coming from a smart bottle and used the data to create event logs. The smart bottle had some temperature sensors and accelerometer. The approach used was to segment a continuous stream of data coming from the sensors using segmentation of the sensor data, then uses feature selection to characterize segments. Once they have the segments, they can interpret the result with the labelled behaviour to finally create an instance process. The main differences with this approach compared to the once we develop is that; 1) they make use of sensor data and not motion data and 2) The goal is to identify activities within the continuous stream of data, whereas in our case the data is already labelled for each activity as that is part of the software provided by KIT-AR.

In [8] we find the most similar to this work where the authors implemented a tool for automatically detecting actions in an assembly workflow. The tool they created uses RGB-cameras and trains an AI model to recognize the activities and label them, falling again into the field of study regarding mapping of activities. In comparison to our approach, this is only validated on a set-up environment and not in real-life scenario.

In another study, the same authors proposed in [14] the authors use a camera to analyse the tasks and movement during a process. The main goal of this study is to produce some data of assembly worker and analyse it through process mining. Another difference is that they use their approach to classify movements and not compare them to each other and detect different work practices. They set up a fixed camera on a workstation and the worker performs some tasks [14], that are recorded by the **RGB-D** camera on top. This however has many limitations: the work is limited to that area, there is only a 2.5D view, and the setup is complicated and hard to replicate.

In addition, the results were not satisfactory and required further work, since the task had many degrees of freedom.

Similarity search in motion data As we need to compare motion data and similarities between them. In [15] the authors wrote an overview of the state-of-the-art methods for motion similarity in 3D human motion. They divide their findings under three sections. 1) Similarity comparison, 2) action recognition and 3) action detection & sub-sequence search operations as can be seen in Figure 2.1.

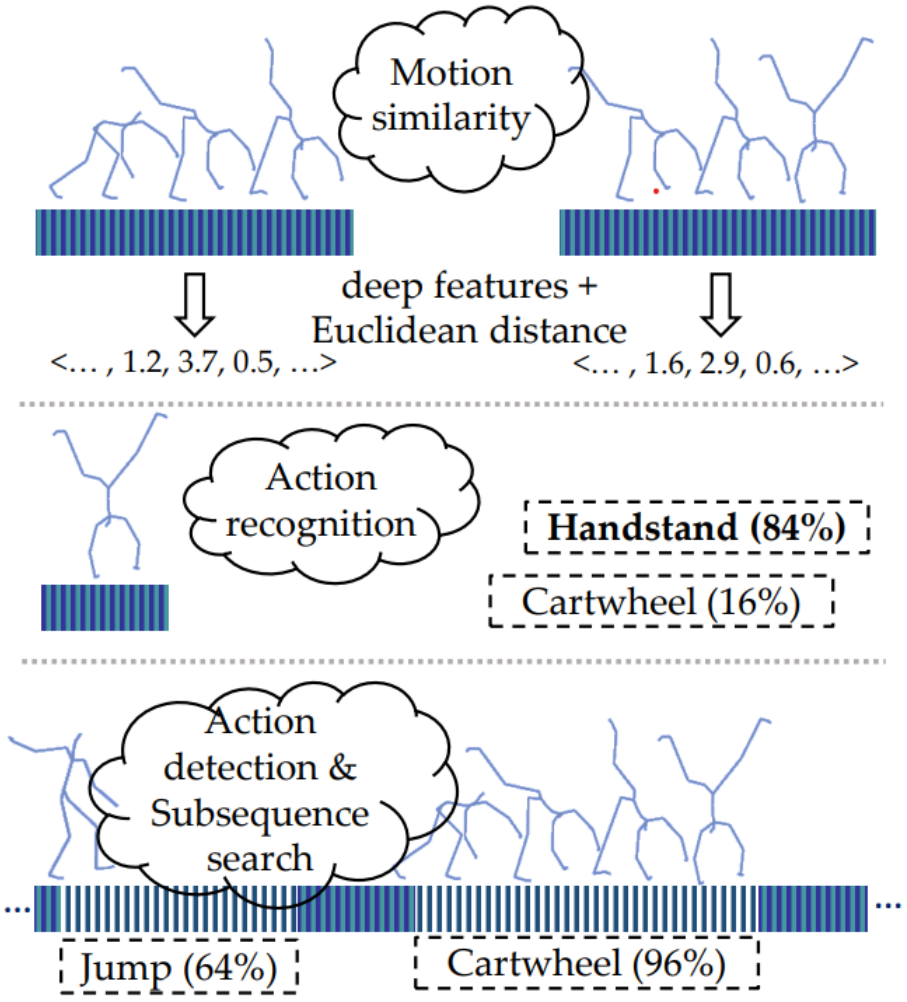


Figure 2.1: Overview of the phases of 3D similarity in motion data. (P. Zezula & J. Sedmidubsky, 2019, p. 1).

Another approach can be found in [16] where the authors proposed an approach to solve the high variation problem and temporal issue is by extracting motion motifs and introduces the concept of motion signatures. They achieve so by using the joint rotation of the motion data and a combination of DTW and a triplet loss convolutional network, achieving good results improving on the results of [17] and [18]. However, this approach is considered weak when considering data with high variability, and they have used only a subset of the dataset, cherry-picking the motions with the best results for the given metric. It uses dance movements and these were known and do not have a lot of change over the dataset. In comparison, our approach is able to work with any type of data in an almost unsupervised way, as long as the data is already labelled.

Grouping and clustering in motion data regarding the clustering of activities, there has not been many approaches using 3D motion data. In [19] the authors proposed the use of unsupervised shapelets [20] to cluster time series, while using DTW to align the time series. This approach however only extracts a feature from the time series and is hard to generalize to 3 dimensions. This in turn was following the approach in [21] but, instead of using statistical features, they use sub-sequences of the data as features. One of the key points of *u-shapelets* is that it can ignore irrelevant data when clustering, works perfectly with datasets with different object lengths, and finally it can have a non-class label where all the non-clustered objects end up, making sure only the similar objects get clustered together.

There has been many attempts at using clustering for geographical data in data mining and machine learning, specifically for the field of geographic knowledge discovery [22] [23].

In [6] the authors proposed an improved space-time clustering approach that relies on agglomerative hierarchical clustering to identify grouping of movement. In addition, they also present a formula that can be generalized with generic distances and attributes. Compared to our approach this study only uses some synthetic data and not on real life data, in addition the distance fiction implemented is simply Euclidean distance, whereas we defined a DTW distance and made use of the ad-hoc distance measure.

Chapter 3

Problem Exposition

The problem this thesis tries to solve is: given a work pattern or trajectory to calculate the similarity between the tasks and, on a case level, to provide the similarity across all cases. This can provide great insight on how each of the subjects execute the tasks and how their work practices differ. The ultimate goal is to be able to cluster activity executions into meaningful clusters that reflect different work practices. After that, looking at it from a process mining standpoint, having this information can help us improve the process, understand why these different work execution, if any, appear in the first place, and ultimately trying to solve them.

3.1 Context/Business Understanding

In workflows with manual tasks that require a human and machine interaction, having control over the human behaviour and its interaction with the system is key for engineers and managers to understand and improve on the work station. This knowledge can be used to construct or improve an assembly work station [8]. In our case for the company KIT-AR, it might be useful to understand how to improve the interaction of the worker with the environment and to make more efficient use of the HoloLens.

We therefore discuss the proposed approach in the context of a car manufacturing process that uses AR glasses to guide the worker. The AR glasses are connected to a system that records and stores all the executions of the given tasks at a certain frequency. The tasks and the steps to follow are shown to the worker via AR technology and can interact with it using their hands, HoloLens can see and map the position of all the finger and hand joints that are visible thanks to the depth camera mounted on the headset (see Figure 1.2). The worker follows the instruction and executes the task

prompted, and then moves forward or backward if there is any problem. The intended workflow is linear and sequential, meaning that the worker should follow the steps prompted on the virtual screen and go backwards only if there is a problem or error.

Each worker has a set of tasks to execute and when that task is finished, the worker ticks a box and performs the next task. A task can vary from opening a container, to moving certain parts Figure 3.1 (a), to making sure all the parts are inside a KIT of parts or making sure the piece is intact Figure 3.1 (b) If there is any problem with any of the tasks, the user will be able to flag it in the AR window.

(a)

(b)

Figure 3.1: Two examples of the KIT-AR workflow, using HoloLens. *(images from kit-ar.com, 2021)*

3.2 Data Understanding

The data used is a modified and anonymized version of the produced data from the worker, containing a list of event points for each of the hand joints and one for the AR glasses that corresponds to the position of the headset, usually on the head. The data points are represented using three-dimensional Cartesian coordinates, where the origin represents the marker that is set by

the worker during the AR setup of the environment and calibration on the KIT itself.

Case	Task	Sensor	Timestamp	x	y	z
0	BarcodeRead1	ARCamera	0	-0.248	1.645	-2.664
0	BarcodeRead1	ARCamera	0.212	-0.245	1.643	-2.664
0	BarcodeRead1	ARCamera	0.447	-0.243	1.644	-2.664
0	BarcodeRead1	ARCamera	0.661	-0.243	1.644	-2.664
0	BarcodeRead1	ARCamera	0.879055	-0.243	1.648	-2.659
0	BarcodeRead1	ARCamera	1.095055	-0.235	1.65	-2.625
0	BarcodeRead1	ARCamera	1.329055	-0.182	1.639	-2.595
0	BarcodeRead1	ARCamera	1.546055	-0.07	1.633	-2.586
0	BarcodeRead1	ARCamera	1.778055	0.111	1.65	-2.6
0	BarcodeRead1	ARCamera	2.011055	0.248	1.687	-2.606

Portion of the raw data from the KIT-AR System

Dataset creation:

Before utilizing the dataset, we need to perform a series of operation to anonymize the data and make it easier and more scalable to read and perform calculation of the data-points. The steps that we'll use are adapted from the work of Alan and Antonio [24] where they performed gesture recognition.

Normalization The first step in order to compare different execution of a task in a three-dimensional setting is to normalize the data, and that is done by finding the average point's coordinate also called a centroid and offsetting every other point by that value. This ensures that the trajectory does not change, only the absolute location of the points will change. This step is explained in detail in section 4.2. **Data type conversion** The second step is to convert some data fields like the case identifier into a number instead of an alphanumerical identifier, speeding up the calculations.

3.3 Detailed Research Questions

Based on the previous premises, the research problem is then divided in two: The first is how to compare these motion trajectories, in three dimensions, without caring about height of the user, scale and time of execution.

The second is to find a method to identify different clusters of executions of cases that would identify the different work practices recorder on the device.

3.4 Detailed Method

The method we will be using to compare the results will be by applying process mining techniques such as the Interactive-data Aware Heuristic to mine a model before performing the clustering. The tool we are using is ProM. We create an event log from the raw data presented in section 3.2 in xes format [25]. Once we have the log, we mine the model visualized as Directly-Follows graph with the most common traces to avoid spaghetti-like model that would be hard to read and interpret the results later on. The filtering we apply is based on frequency, meaning that for the graph we consider the activities happening at least a certain amount of times, this is done using the filter function in ProM.

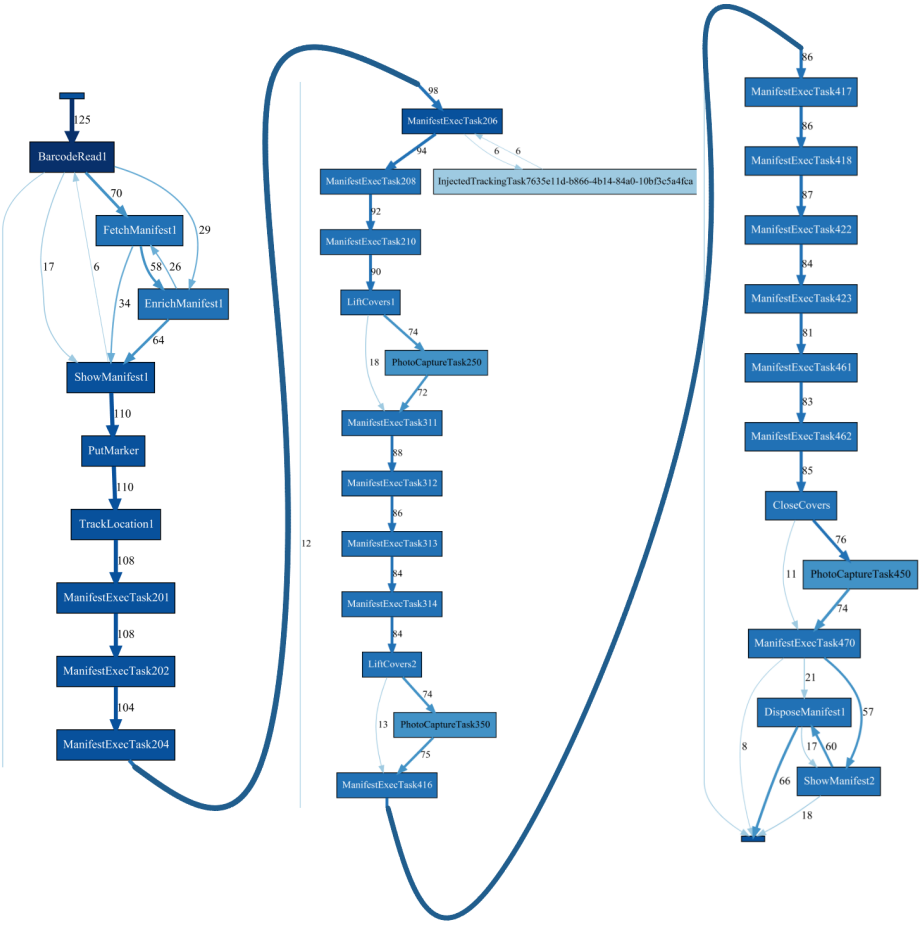
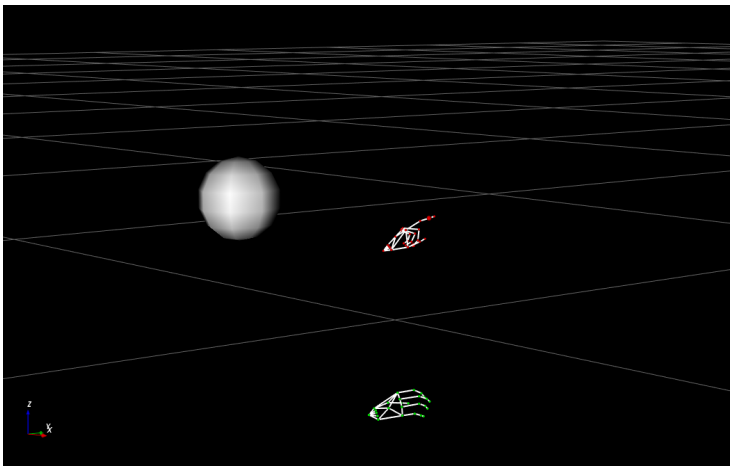


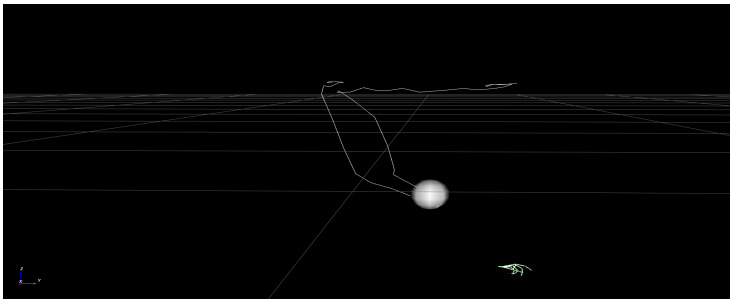
Figure 3.2: Mined DFG from the data, without any clustering

We use this model to compare it with the resulting model after applying the clustering, having K new models, one for each of the clusters as an output of the method introduced in section 5.2.

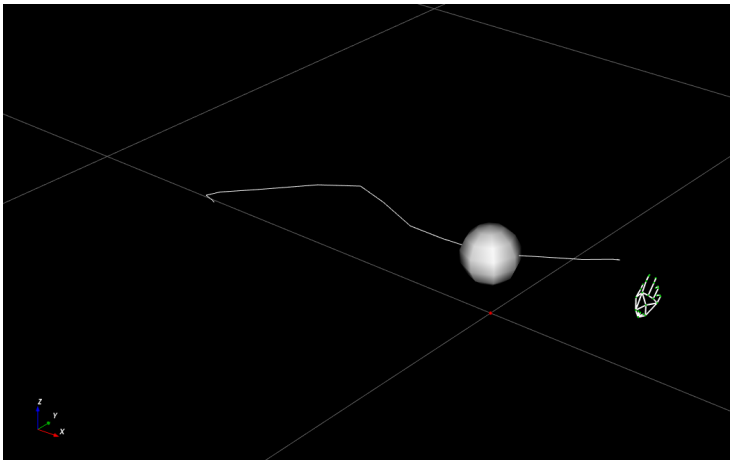
We also use Mokka tool [12] to visualize certain trajectories of the motion data converted into c3d format [11] of the various clusters and domain knowledge from inside the company to assess the validity of our results. Some examples of the visualization of the tasks in Mokka is shown in Figure 3.3. Inside this tool we are able to assign different colours (Figure 3.3 a), different sizes and visualize the segments between joints. This can help us visualize better the head location, given by the location of the Headset and hands location. Finally, it allows to visualize the trajectory (Figure 3.3 b and d) of a certain joint.



(a) BarcodeRead



(b) Manifest201



(c) Close Covers

Figure 3.3: Different example of activity visualization using Mokka.

Chapter 4

Comparing different trajectory

In order to compare two different executions of a certain task and in order to do it in three dimensions, we make use of Dynamic Time warping [26] that has been the de facto standard for time series analysis. This is very useful for the scenario we have and specifically to compare executions from different actors, with different heights, body form and different speed. Another relevant characteristic of Dynamic Time Warping is that it provides a warping path, meaning that given two time series, the algorithm will align them together and provide a path that minimizes the distance difference. This concept is particularly useful in the case of subtask executions where we want to see not only the distance between two given trajectories but also how one deviates with another. The distance provided by the algorithm will then be used in Chapter 5 together with other metrics to cluster activities. This measure will represent the similarity in motion.

4.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a popular distance measure introduced first by the authors of [4], initially for the purpose of speech text recognition, but soon this method became widely used in many other fields [27] [28].

The algorithm uses two trajectories as input and matches the vertices from one trajectory to the vertices of the other such that the summed distance is minimized. This means that the algorithm is able to align two trajectories to find the optimal match [29]. Since there are many paths between the reference and the test trajectory, therefore the optimal warping path is the one that minimizes the distance.

$$d_T = min \sum_{i=1}^{N} d(T_r(w(i)), T_t(w(i))) \tag{4.1}$$

In the equation above $w(i)$ represent a single *warping path* and d is a distance function, typically Euclidean distance, used to compare the two points.

For example, given two one-dimensional trajectories T_t and T_r defined below:

$$\begin{aligned} T_t[n] &= [0, 0, 1, 4, 3, 1, 0, 1, 2, 2, 1, 0, 0, 0, 3] \\ T_r[n] &= [0, 1, 2, 3, 1, 0, 0, 0, 2, 1, 0, 0, 1] \end{aligned} \tag{4.2}$$

Even though the two series different in length and in values, when we apply the DTW algorithm we find the warping path between them and, is then able to find the similarities between them, as shown in Figure 4.1.

The algorithm works in a simple way, instead of checking for distances in the trajectory and select the warping path (or matching path) based on the shortest distance of the current point, it computes a matrix of distances based on current point to the neighbouring points, the proceeds to compute the best path, that goes through the shortest values at any given point, until reaching the end, where the computed distance is also the distance between the two trajectories.

The images show two trajectories that are being compared to each other. When a point in the same position does not match the other trajectory, the neighbouring points are checked and the shortest distance is take, based on Euclidean distance.

4.2 Data Preprocessing

Before utilizing the dataset, we need to perform a series of operation to anonymize the data and make it easier and more scalable to read and perform calculation of the data-points. Including the normalization and converting the case ID into a sequential number instead of the alphanumerical identifier, this way we the operation can be speeded up.

The steps that we'll use are adapted from the work in [24] where they performed gesture recognition.

Normalization The first step in order to compare different execution of a task in a three-dimensional setting is to normalize the data, and that is done by finding the average point's coordinate, also called a centroid, and offsetting every other point by that value. This ensures that the trajectory

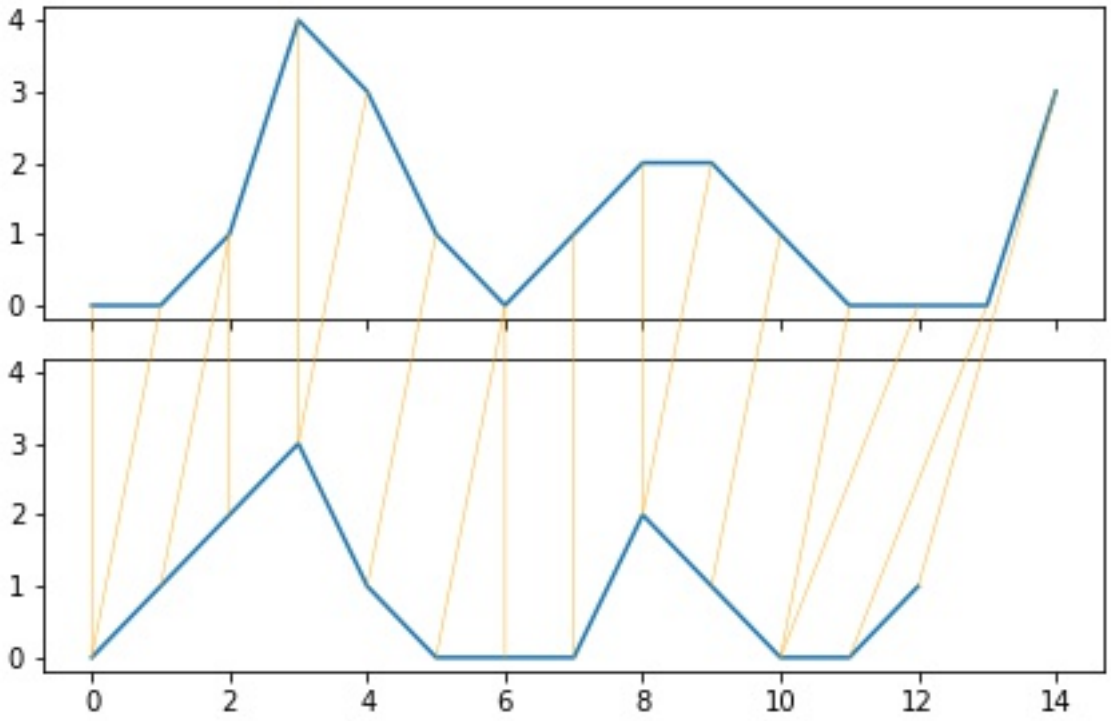


Figure 4.1: Visualization of alignment using DTW on the two example series

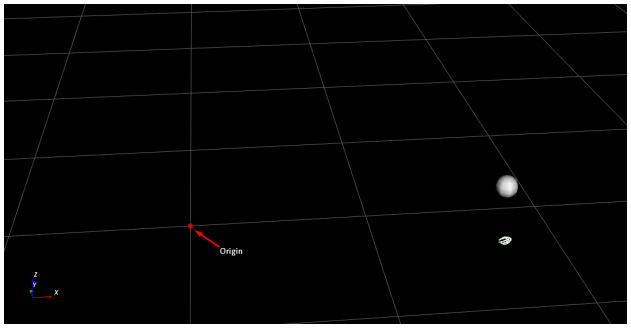
does not change, only the absolute location of the points will change as shown in Formula 4.3.

$$c' = (\bar{x}, \bar{y}, \bar{z}) = \frac{\sum_{i=1}^{n}(x_i, y_i, z_i)}{n} \tag{4.3}$$

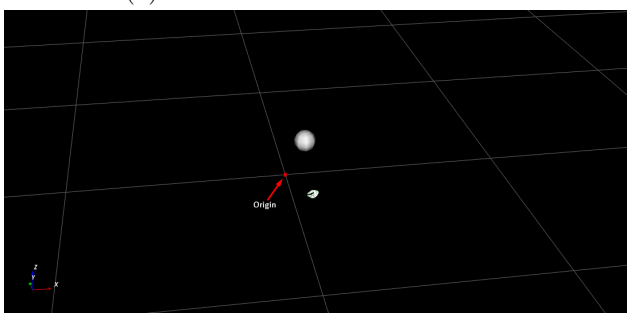
Once we have the centroid, we subtract from each point of the trajectory the value of centroid on each of the axis (4.4).

$$(x_i, y_i, z_i)' = (x_i - \bar{x}, y_i - \bar{y}, z_i - \bar{z}) \tag{4.4}$$

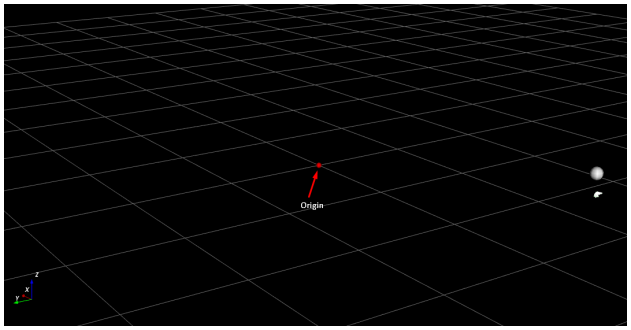
This helps the DTW algorithm to compare two trajectories despite being different in location and scale. A result of the normalization is shown in Figure 4.2. It is possible to see how normalizing our trajectories can be helpful, since for the same activity the location of the joints is not always the same. For example, in Figure 4.2 a and c we see that the position of the headset and the location of the hand are very far from the origin, but after applying the normalization (Figure 4.2 b and d) we can see that they come closer to the origin.



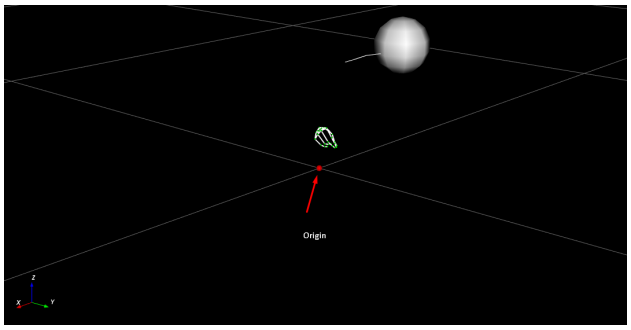
(a) Case 10 before normalization



(b) Case 10 after normalization



(c) Case 15 before normalization



(d) Case 15 after normalization

Figure 4.2: Effects of applying the normalization **FetchManifest** task in KIT-AR dataset, visualized using Mokka

4.3 Multidimensional DTW

In order to use the DTW for our 3 dimensional data, we need to use multidimensional Dynamic Time Warping. Using this algorithm, we can compare two given trajectories on the three dimensions and then merging the result. However, the way it is done can change the result of the calculation by a lot, mismatching the class. The reason this happens is that the two ways to calculate the multidimensional DTW use different approaches to merge the three dimensions. As a consequence, choosing a different approach can give a completely different result.

Therefore, we rely on the analysis and discussion in [26] about the importance of choosing the right method to apply. There are two main ways of to generalize the classic DTW approach into a multidimensional case: Figure 4.3 shows a simple example demonstrating the difference in the result.

4.3.1 DTW-I

DTW-I is the distance of each dimension measured *independently* under DTW added to each other. Meaning that the distance of the three-dimensional series would be the sum of the DTW on series of x, y and z.

This means that each dimension is considered separately and independent of the others.

The calculation of the DTW-I is shown in the equation below, where i is the dimension considered.

$$DTWI(T_1, T_2) = \sum_{i=1}^{N} DTW(T_{1_i}, T_{2_i}) \tag{4.5}$$

4.3.2 DTW-D

The second way to calculate Multidimensional-DTW is by using dependent DTW or DTW-D. Instead of considering each dimension separately, we consider the dimensions in an n-dimentional space and calculating the distance between the two points.

The equation for the DTW-D becomes the following considering two points instead of trajectories as a whole, redefining the distance function d as the cumulative Squared Euclidean distance, where **n** is the number of points being considered.

$$DTWD(p_1, p_2) = \sum_{i=1}^{N} d(p_{1_n} - p_{2_n})^2 \tag{4.6}$$

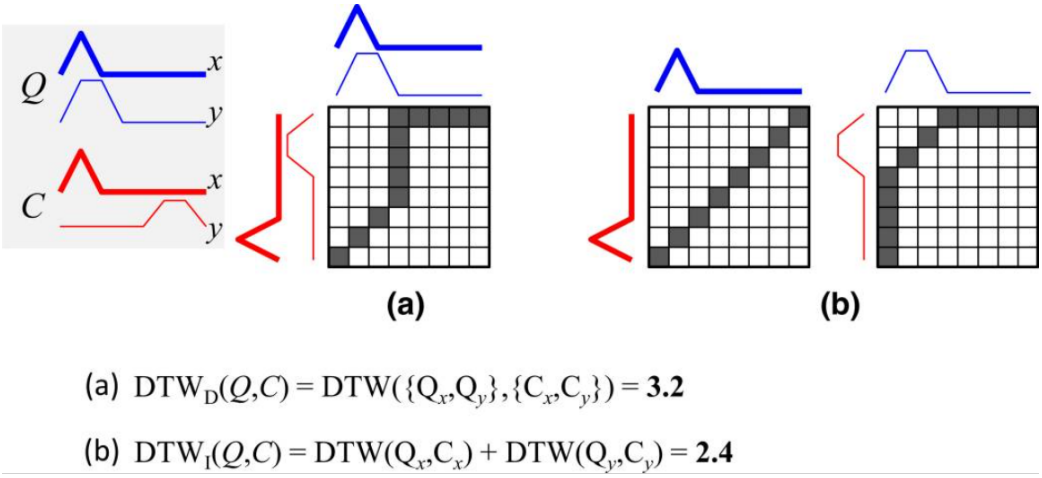


Figure 4.3: top left: Two multi-dimensional time series. a) The Dependent DTW distance between them is 3.2. b The Independent DTW distance between them is 2.4. *(Shokoohi-Yekta et. al., 2016,p. 3)*

4.3.3 Comparison and choice

After introducing the two different implementation of the algorithm, we now compare the two for our specific case and decide which of the two methods will fit our data and why.

Based on the implementation and the design, the question is ultimately, "*is our data on one dimension dependent on the other dimension?*" since that is the main difference of the two. *DTW-I* considers all dimensions separately, whereas the DTW-D considers them in a dimensional setting.

In [30] the authors argue that since the data from their domain (satellite image data) have dependent dimensions, it is wise to use DTW-D.

Following that argumentation and since the data that we are considering is motion data, we assume that there is high correlation between the three dimensions, and therefore we will use the DTW-D.

Although we decided to go with one method, both approaches were considered and implemented.

4.3.4 Implementation

After choosing the approach, we integrate it in the algorithm implementation from [31] library that comes with pre implemented functions for DTW and multidimensional DTW.

The algorithm computes a distance matrix based on Formula 4.6 that then can be displayed using a heat-map, and on each axis we visualize the 3

trajectories, one for each dimension. In Figure 4.4 we can see the result of this computation on two very similar activities from the Karate Dataset [5].

As can be seen from the optimal DTW path (in red), that is mostly diagonal, meaning the two trajectories are very similar. In the dark blue areas the value of the distance is lowest, therefore the path follows those values as per design, whereas close to the top-right corner and top-left corner the values of the distance is higher since we are comparing the beginning of the trajectory with the rest of it.

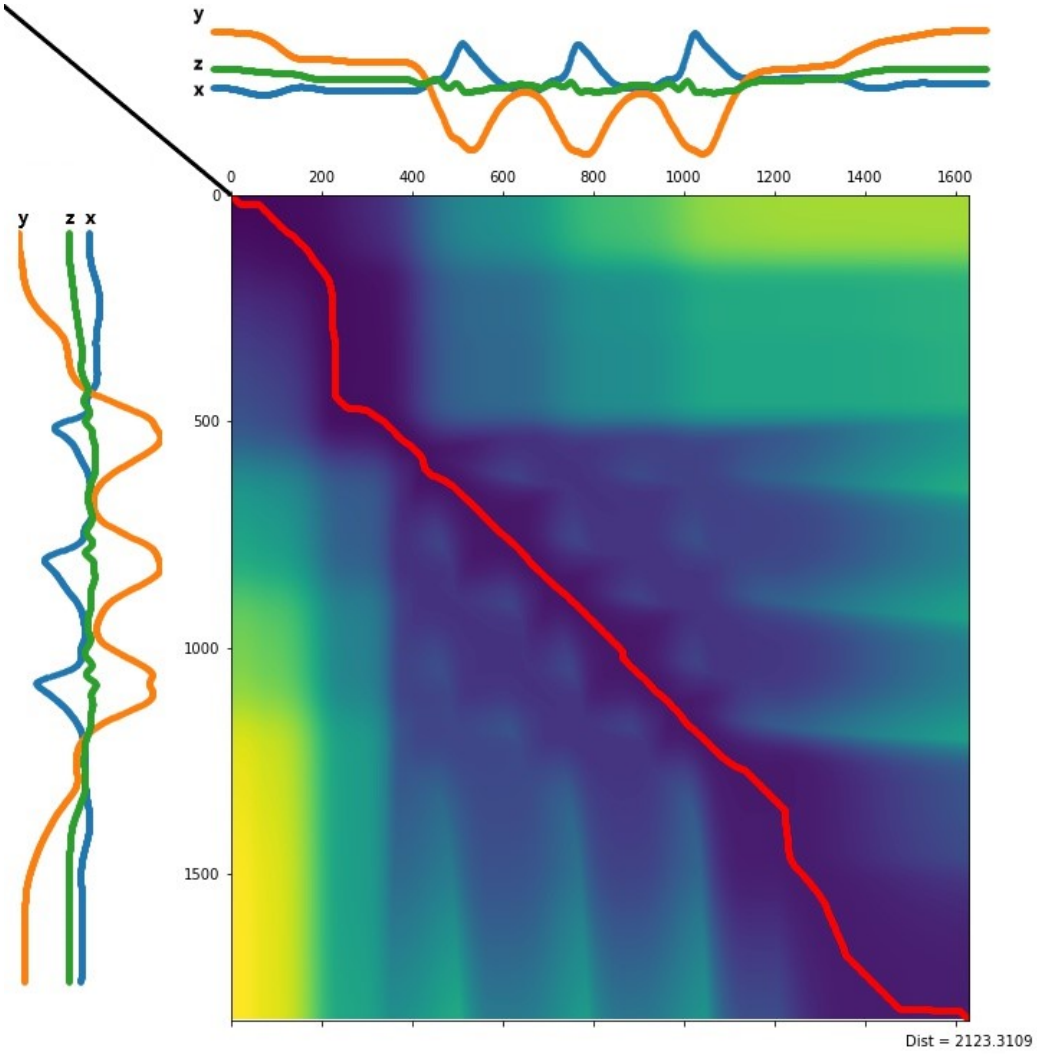


Figure 4.4: Two multi-dimensional time series of a technique execution from Karate Dataset

At the end of this process we will have a DTW distance, that corresponds

to the value of the last cell, this represents how much the path deviated and the cost of such deviation based on the distance. The value of the distance will be used later to compute the pairwise distance between all the trajectories.

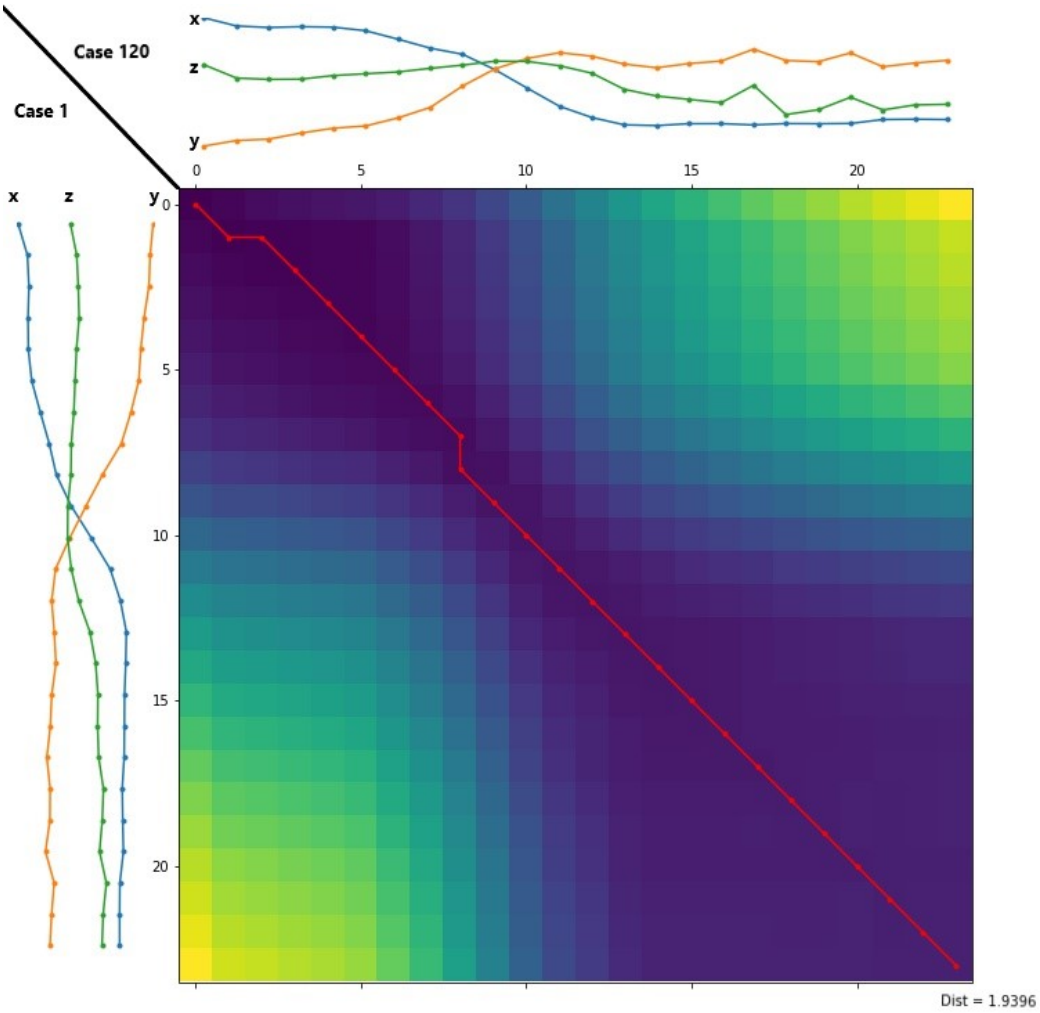


Figure 4.5: Comparison of the task Close Covers in cases 120 and 1

In Figure 4.5 we can see the two trajectories composed of the three dimensions in blue, orange and green, respectively x y and z coordinates. In the bottom right corner we can see the warping distance, that indicates how much the two trajectories differ, the higher the distance the more different they are, when applying twice the same trajectory we obtain zero as a result, since two identical trajectories do not differ at all.

In Figure 4.5 and Figure 4.4 we can see the similarity matrix coded in colours. The lighter the colour, the higher the distance between those two

points. As we can see in the bottom left and top right corners the colour is the lighter reaching a yellow colour at the extremes, this is because we are comparing the beginning of one trajectory to all the trajectory and the distance inevitably increases. On the other hand, if we keep following the diagonal of the matrix we can see the comparison of the single points, if the trajectories are identical then the best warping path (in red) should follow exactly the diagonal, meaning that each point corresponds to an identical point on the other edge. The closer to a diagonal the warping path, the more similar the two trajectories are.

When comparing the two figures from the two datasets, it is easy to see that one has far more data points and granularity than the other. This because the tasks are short in the case of KIT-AR and only require little movement in the majority of the cases.

Chapter 5

Clustering of activities

Finding clusters is an important task in many spatial analyses tasks, in order to explore the trajectory and to find insight on the spatial data. For the scope of this thesis, we need to cluster activities and retrieve the different work practices by leveraging the spatial data, temporal data and other attributes that can be useful from a process mining perspective.

The requirement of such algorithm are that it needs to be:
Generalizable: the algorithm needs to be generalizable to any type of data and any domain, as long as it has motion data in it and some process information.
Flexible: It needs to allow for different weights to the different metrics based on the task or question it aims to answer. For example, if the temporal metric is more important to analyse, but we still want to take into account rest we can give more weight to the temporal dimension and lower values to the other metrics.

5.1 Spatio-temporal Clustering

Based on these requirements just defined, we make use of the approach described by the authors of [6]. In this paper, they introduce an improved spatio-temporal clustering algorithm that makes use of **general** and specific distances and is purposefully generic to accommodate any domain. This is done by using general distances and attributes that can be specific to the domain.

The equation below is the adapted equation used for our specific case.

$$D(c_i, c_j) = w_1 * \frac{DTW(\bar{x_i}, \bar{x_y})}{S_{DTW}} + w_2 * \frac{D_t(\bar{t_i}, \bar{t_y})}{S_{D_t}} + w_3 * \frac{LD(\bar{p_i}, \bar{p_y})}{S_{LD}} \quad (5.1)$$

5.1.1 Implementation

Equation 5.1 is composed by a set of metrics and weights assigned to them. After that, the result is standardized or scaled using the S term for each metric. Once we have computed the similarity between two activities, we can start clustering them to retrieve similarity in the execution of cases at a macro-level, using notions calculated in the previous chapters.

After we define the trajectory/trajectories for each task, and we define the sensibility for each task using the weight $w0...w3$. We can compare even in real time the trajectory and output a similarity measure. This similarity measure will characterize how similar the movements in the trajectories of the case are, and how these two cases compare to each other from the other metrics.

The equation 5.1 is composed by a series of general distance measures like the DTW and Time difference, as well as more specific attribute distances and Levenshtein Distance. These set of distances will be explained in detail in the following sections.

The clustering algorithm and the DTW comparison have both been implemented in a Notebook using Pyhton 3.6. For easier computation and change, we save each of the calculation on disk, so that we do not have to run it again. For the DTW part we make use of a set of publicly available libraries such as dtaidistance [31] and Matlab plot Library. For the clustering, we use Sklearn, with the `metric="precomputed"` argument in the functions of the Silhouette and the Agglomerative clustering. The reason for this that we already computed the distances between observations in a similarity matrix w.

5.1.2 Dynamic Time Warping distance

The first distance measure that we defined as introduced in section 4.1 is the DTW to compare the two tasks execution in a spatial manner, using this distance we can track similar execution of a task or a set of tasks and retrieve information that would never be possible with classic Process Mining techniques.

5.1.3 Temporal distance

The second distance measure is a simple temporal comparison in the execution of a task. This is helpful to extract the temporal dimension since the DTW aligns the tasks even if they happen at different speed, however our

goal is to compare both and, based on the scenario, give each one a different weight.

5.1.4 Levenshtein Distance

Finally, to introduce process mining measures, we use a modified version of the Levenshtein distance [13]. This will help us compare the flow of two cases, the order of the activities, if a task/activity is missing or if there is more than one.

5.1.5 Quaternion DTW

The quaternion DTW is a special variant of the DTW that instead of using Euclidean distance, it uses quaternions distance. This data is coming from the AR glasses and tells us at any given time when the person is looking or where the Glasses are directed at. For this implementation, [29] definition was followed and implemented in Python [32].

This measure was added since it was interesting to plug another distance measure into the DTW showcasing that it's feasible to have multiple metrics for any given domain. However, in the final clustering algorithm, it was not used. This is because we deem it adds complexity to the algorithm, and for this first iteration we keep it simple.

5.2 Clustering Values

After defining all the metrics we are using, we put them together using the equation 5.1.

The result of this step will be a distance that takes into account all these metrics and add them together. Resulting in a matrix of pairwise distances as an output. An example of the result for DTW distance calculation is shown below in Table 5.1. As we can see the values on the diagonal show a value of 0, this is to be expected since the motion data distance is the same and therefore DTW-D algorithm returns a value of zero.

Another important observation is that the DTW distance value between cases i with j is equal to comparing cases j with i. In the implementation, we check for the one with the longest sequence, and we use that as a reference. This implementation decision allows us to run the script on half of the matrix reducing drastically the execution time of the script.

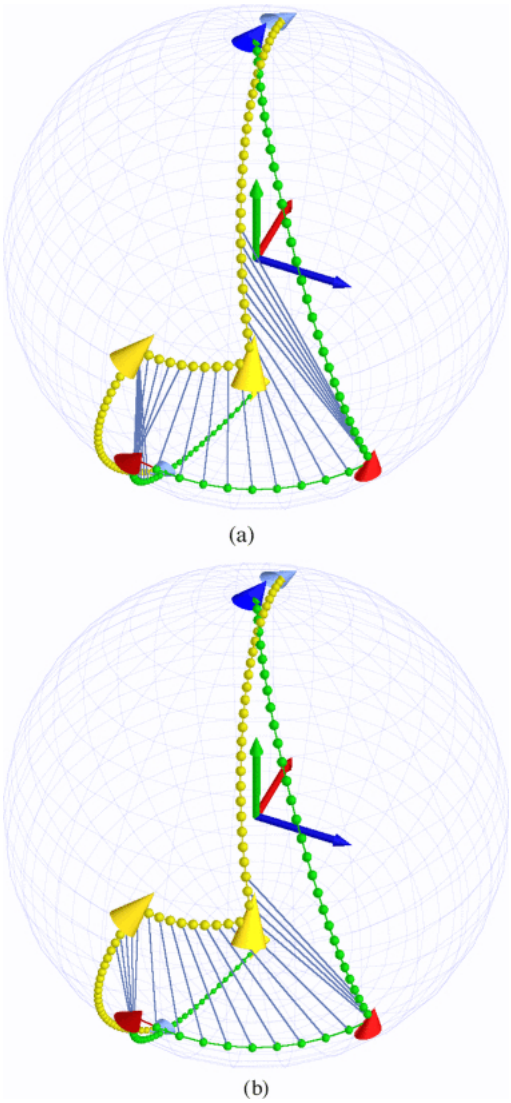


Figure 5.1: Quaternion Dynamic Time Warping *(B. Jablonski, 2012, p. 5)*

case Id	**1**	**2**	**3**	**4**	**5**	**6**
1	0	228.71	8.54	7.50	258.6	76.5
2	228.71	0	26.13	15.55	269.3	96.35
3	8.54	26.13	0	2.69	1.88	18.74
4	7.50	15.55	2.69	0	3.01	18.91
5	258.6	269.3	1.88	3.01	0	74.87
6	76.5	96.35	18.74	18.91	74.87	0

Table 5.1: Portion of table for the DTW-D calculation

Chapter 6

Evaluation

In this chapter, we evaluate our findings using the Silhouette method, which is a common score when using DTW [33] [27] [28]. Following the introduction to the silhouette method, we introduce the setup and the methods used for this evaluation. Finally, in the discussion section we analyse the results of the clustering and the working practices identified, for each of the clusters we plot DFG and in what they differ with each other. We include interviews from the domain expert about the results and the possible causes for such clustering.

6.1 Silhouette method

After the distances are calculated we need to decide how to compute them together and specifically, the goal is to decide the values of the weights w_i and the optimal value for k, the number of clusters. To do so, we make use of the silhouette method, first introduced in [7]. This is a non-parametric measure of cluster separation where for each case i, the silhouette index is defined as $s(i) = (b_i - a_i)/max(a_i, b_i)$, where a_i is the average dissimilarity between i and the other cases of the same cluster, and b_i is the minimum dissimilarity between i and all the cases of a different type. Based on the value of the silhouette score, we can assess how well that k value fits the data, meaning how well the clustering is, if the score is low it means that the clustering is not very effective, if the score is high it means that the algorithm found good clusters that are all far from each others.

We use the same approach by keeping the k value constant and changing the values of the weights until the silhouette score stopped improving.

6.2 Objective

The objective of this approach is dual: The first is to decide on which value to use for K that can better identify the clusters. By running the clustering function multiple times with different k values and choose the one with the best silhouette score.

The second is to decide on which values to use for weights for each of the distances that make the silhouette score the highest, meaning that all the cluster found are at the optimal distance.

6.3 Setup

The experiment runs as follows: after preprocessing the data we run the pairwise comparison of all the tasks within the case, if a certain task is not present we skip it. This will be accounted for by the Levenshtein distance. After we calculate the DTW-D for all the cases, that in turn are composed by the sum of the DTW of each task within the case.

After having all the matrix distance of the three metrics that we defined, we can compute them together into a final matrix that is the weighted standardized sum of all the metrics. To determine the value of K and the weights, we run the Agglomerative clustering into 4 for loops. Three for the weights and one for the number of cluster. We compute the clusters and the silhouette values inside the inner loop. After that, we store the values in a Pandas data frame on disk, this way it is easy to replicate the experiment at any stage without the need to run the algorithm from beginning, this also helps for the DTW since it requires a lot of time to compute all the distances for the dataset and therefore having the results saved is advantageous.

6.4 Results

One of the best values for the weights are: $w_o = 3, w_1 = 6, w_2 = 1$ and for k = 3. After applying the values to the spatio-temporal formula, we obtain the silhouette plot in Figure 6.1.

After obtaining a K value and the weights with high silhouette values, we proceeded to check and see the results of the agglomerative clustering.

In Figure 6.1 we can see clearly the clusters. One of the clusters is not visible since it is only one case.

Regarding the time difference, we present below the average time of execution of each cluster in Table 6.1. We use cluster 1 as a reference for the time difference since it is the most behaviour, having the most cases. As we

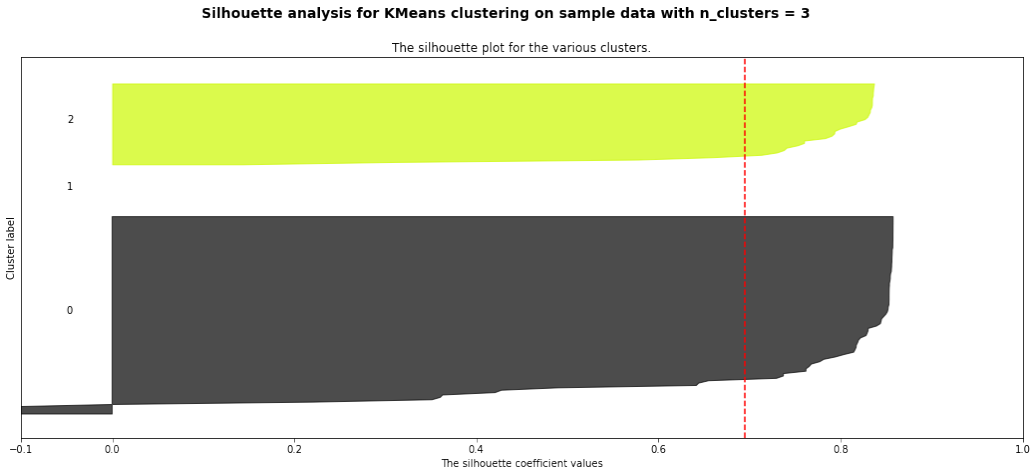


Figure 6.1: Silhouette plot with k=3

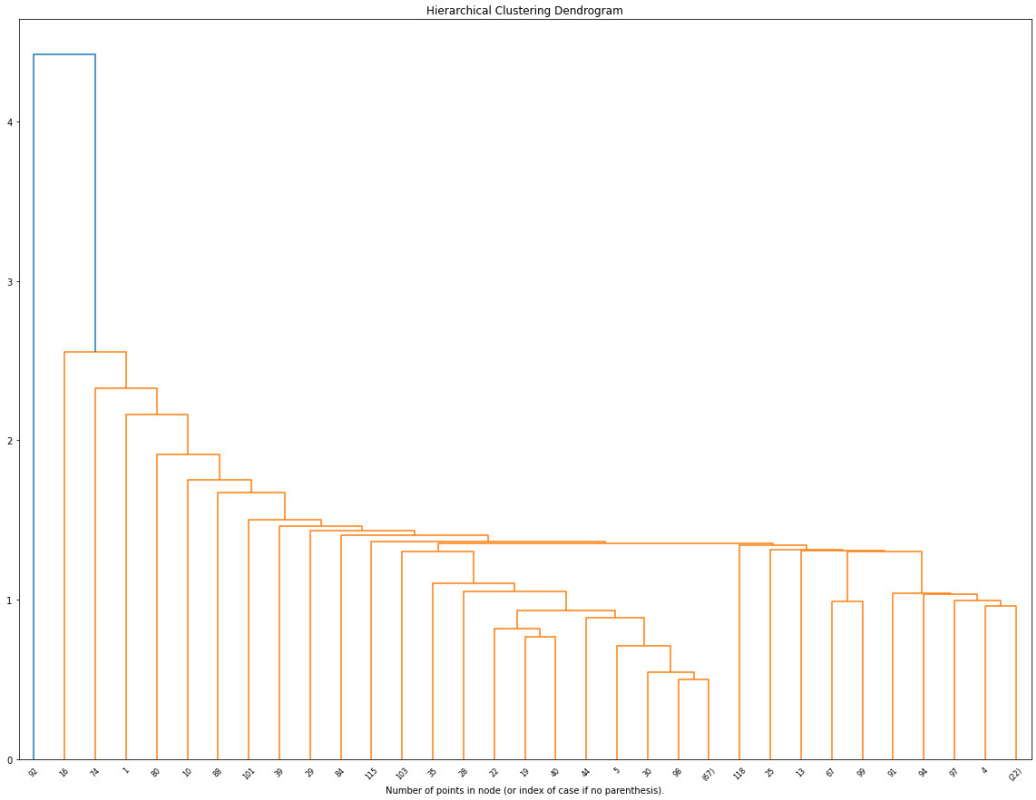


Figure 6.2: LinkageThree of the cases showing the 3 clusters. *X-axis shows the case Id or in parentheses the number of cases in that node*

Cluster	**Average time difference compared to cluster 1 (%)**
Cluster1	-
Cluster2	26% slower
Cluster3(case 92)	68% slower

Table 6.1: Time difference in percentages compared to cluster 1

can see the time difference also reflects the clustering, in fact all the cluster differ with each other, this is to be expected since one of the metrics used is temporal difference.

6.5 Discussion

After clustering the cases, we analyse the results. In the case of our particular configuration, we have mainly two clusters of executions. Cluster with the normal path and a cluster with few cases. Finally, there is an outlier cluster that has some tasks that makes it an outlier. This is case **92**.

Once we retrieve case identifies belonging to the clusters, we filter the data on ProM to mine three models, one for each of the clusters. The results are shown below.

Cluster 1: The main behaviour

The first cluster of execution that the model identified is composed of the majority of cases showing the "normal" behaviour of the model.

An image of the mined model is shown in Figure 6.3, as we can see the behaviour of the model is mostly linear, this reflects our expectations of the model. Where the majority of cases follow a straight path and a portion deviates in specific points. In this case, we can see that the deviation happens specifically on PhotoCapture tasks.

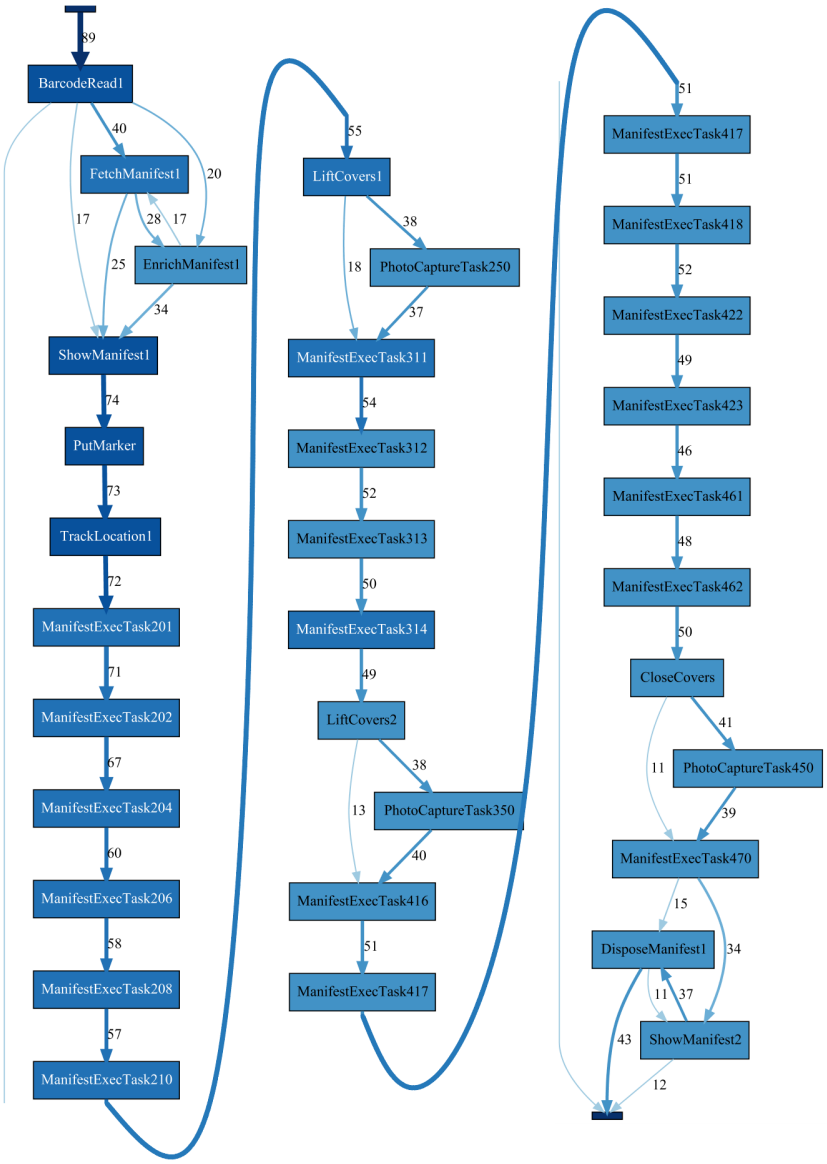


Figure 6.3: Mined DFG for the cluster 1. *(The image has been cropped and joined together for better visualization)*

Cluster 2: Modified behaviour showing on Photo Captured task

This second cluster of work practice shows a slightly different behaviour compared to the first, here we see *Photo Capture* task being more linear with the rest compared to the first identified cluster.

"This task refers to when workers are asked to take a picture of the kit/cart to document what parts are present and which ones are missing or have problems." This was the clarification of the domain expert about this specific task.

In order to investigate this phenomenon, we visualized this as seen in Figure 6.4 using the Mokka tool and by providing the case ID of the cases in the cluster and PhotoCapture as task identifier. We found that in only 6 cases out of 67 there was presence of the **ARCamera**. This is the sensor that tracks the position of the HoloLens itself. Meaning that in more than 91% of the cases in the identified cluster were not tracking the headset. However, the rest of the hand joint were still visible, indicating that there might have been some errors in the reading.

Comparing this to the previous cluster where in all the cases for the task **PhotoCapture** there was a reading on ARCamera, equivalent to 100% of the cases.

After asking the domain expert in the company about this anomaly, this was his response;

"This is a very interesting finding for our internal system diagnostics. We are using the camera to take an image, but the camera is also our source for the trajectory data. So, what may be is that when using the camera to take the photo, the sensor does not provide the position and rotation data any more."

This explains why in most of these cases there was no reading for the headset location, contributing to clustering them together.

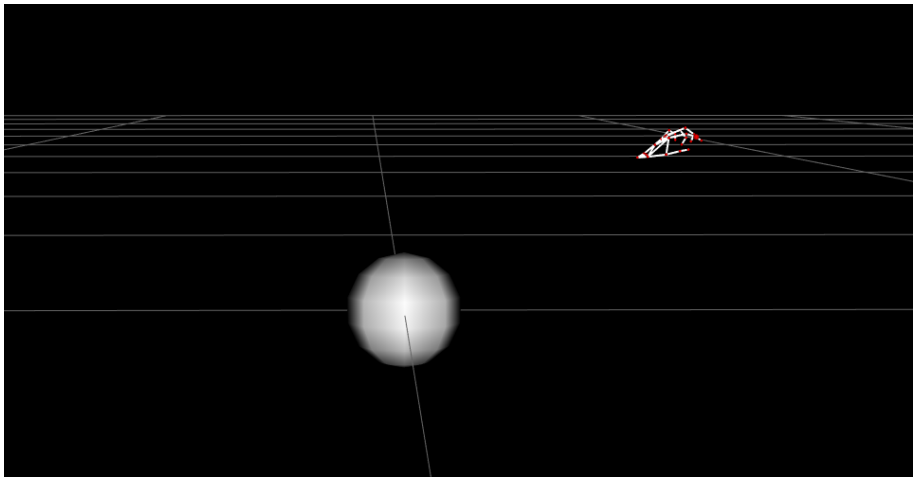
Cluster 3: Anomaly case

The final cluster identified by the model After exporting the log to ProM [25] we found that an activity: **InjectedTrackingTask7635e11d-b866-4b14-84a0-10bf3c5a4fca**

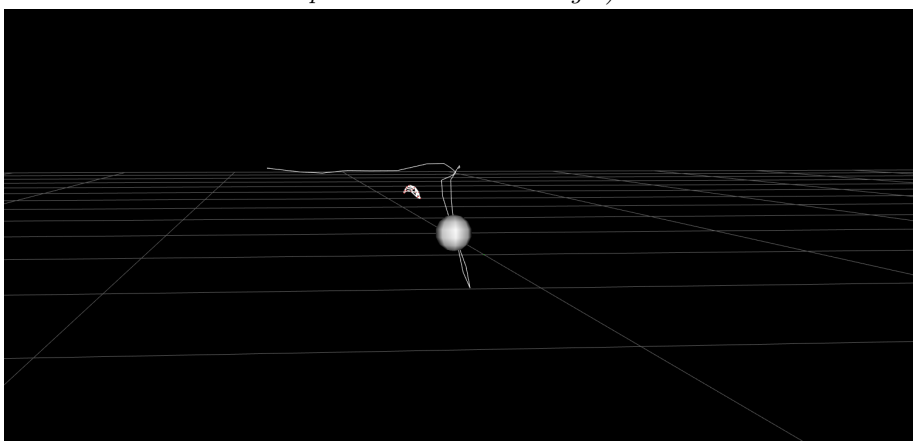
When we run the trace of the case in ProM and plot it with Data-Aware heuristic miner, we have the DFG in Figure 6.7. As we can see, this DFG is different from the other clusters, and the linear structure seen in cluster 1 and 2 is no longer present.

Although it is only one case, we use the same method of plotting for easier comparison with the other Directly Follows Graphs.

After asking the domain expert inside the company for more information of what might be the causes of this behaviour and what exactly does this task do. This was his response.



(a) Case in the cluster where there was no reading of the Headset *(head sensor position still in the origin)*



(b) Case in cluster where the headset joint was visible

Figure 6.4: Using Mokka tool to inspect the abnormal behaviour for *Photo-capture*, where in the majority of cases the ARCamera was not read

"The task is triggered when the worker performs a so-called realignment task. This is usually done when, for some reason, the AR instruction are not well aligned any more with the real-life position that they should be on. By activating this task, which you can think of as a work instruction independent system task, the worker gets the opportunity to look at the QR code marker again to improve the alignment. The reason that it has a strange name with a Universally unique Identifier (UUID) is a purely technical one and has no meaning."

This task therefore indicates that there has been a problem and in this specific case there has been so many, that is why the algorithm identified it

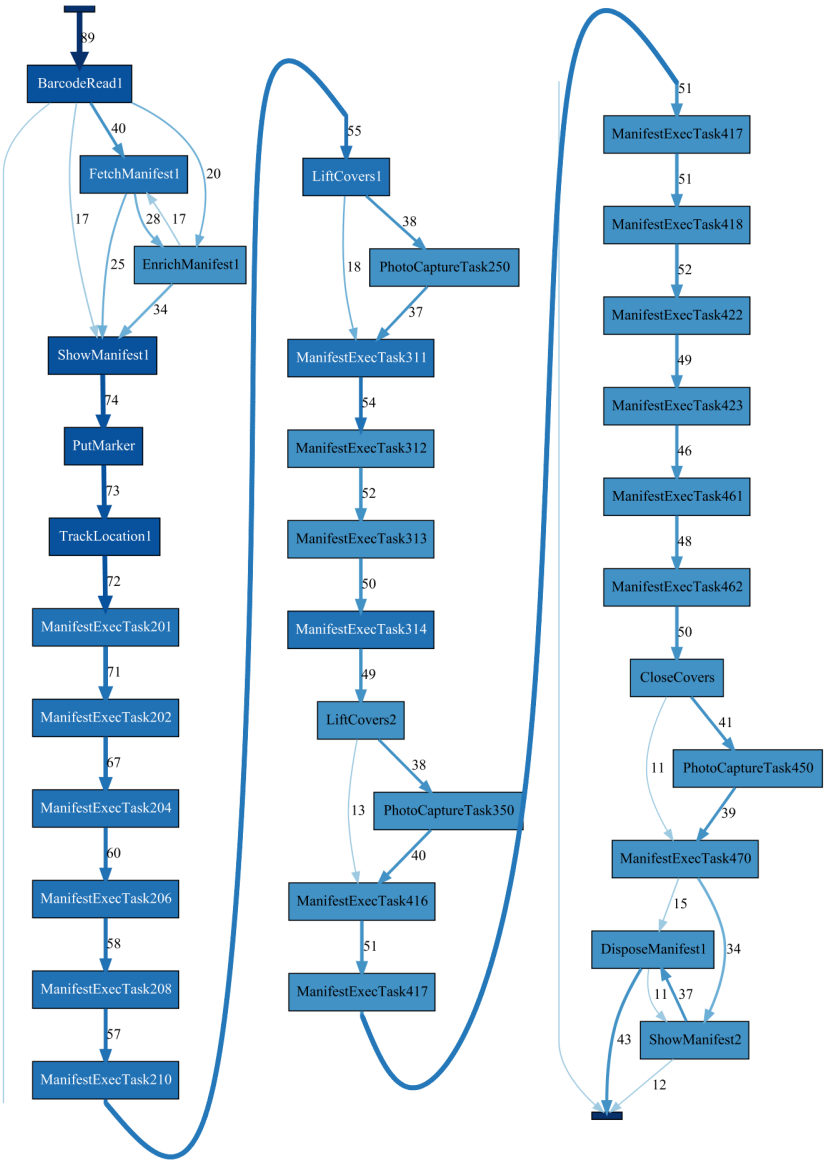


Figure 6.5: Mined DFG for cluster 2. *The image has been cropped and joined together for better visualization*

as an anomaly, because both from a process mining point of view and from a trajectory point of view it did not fit to any of the other tasks.

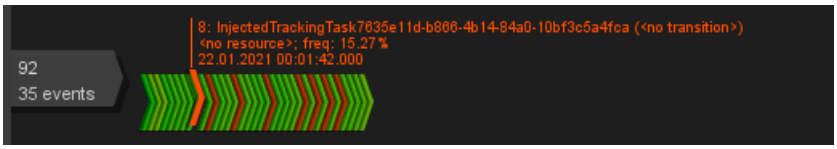


Figure 6.6: Anomalous case discovered from clustering

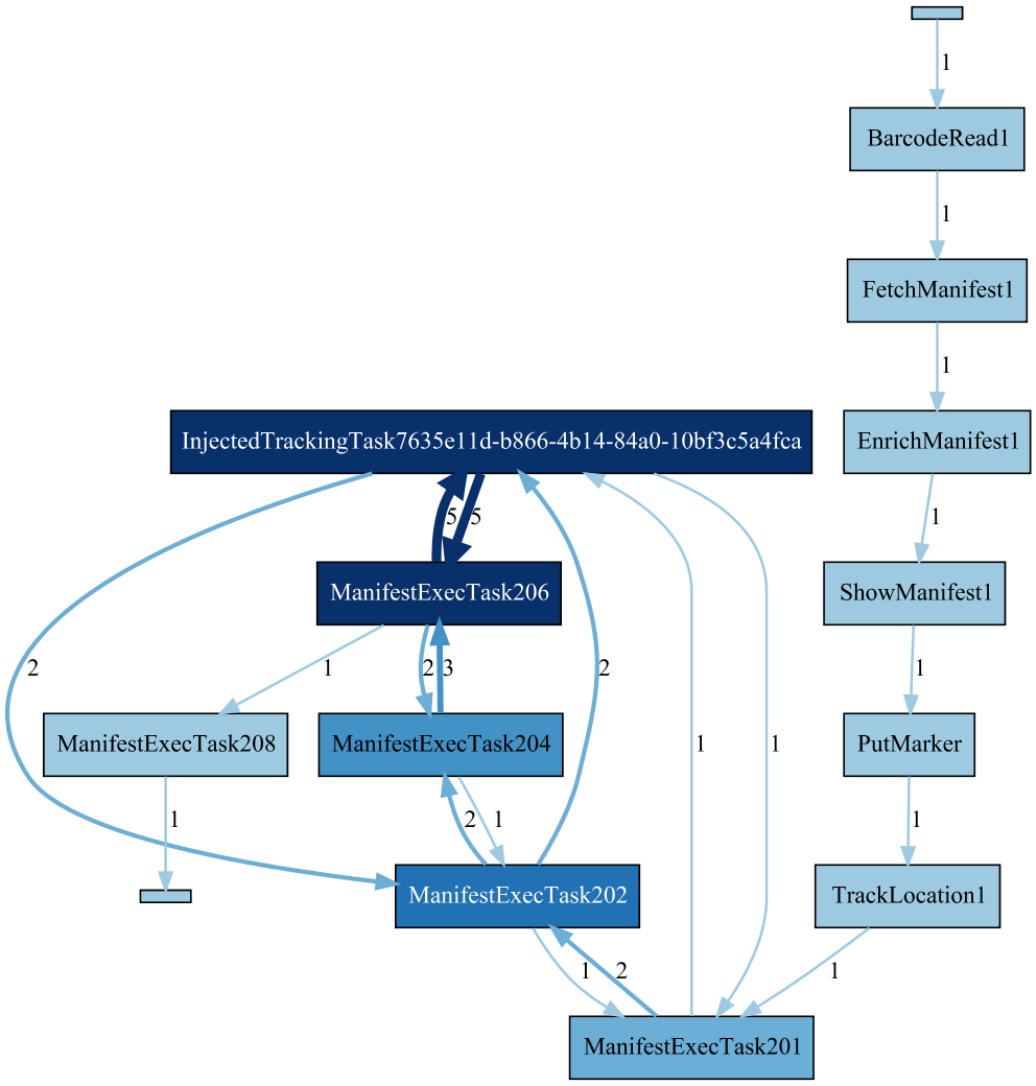


Figure 6.7: Anomalous case 92 DFG

Chapter 7

Conclusion

In this thesis, we proposed a spatio-temporal clustering based on motion data from sensors.

The work is divided into two phases. In the first phase, we compared trajectories of execution using DTW and output a distance measure.

In the second phase we use the outputted distance and, together with attributes, we compute a spatio-temporal distance function used to compute the pairwise distance between each case, and ultimately we clustered the cases using Agglomerative Clustering with the precomputed distance matrix.

For the evaluation and to decide on the weight we used the Silhouette method, and based on the score we tweaked the parameters and the number of clusters.

After applying this method to the KIT-AR dataset, we retrieved the clusters that were used to identify work practices.

We identified a total of three clusters, two with work executions, one of which with relatively fewer activities and one with the rest of the activities.

To explore the clusters, we mined the process model using ProM and produced event logs from the dataset using P4PM for Python.

The main limitations of our approach are:
Validation: This is rather a weakness of the dataset used, since we did not have any ground truth it was hard to test the validity of our approaches and many times we either had to use a second dataset to validate or refer to the domain expert to analyse the results.
Computational: Using the DTW is very handy for our approach, but comes with limitations. One of which is the computational speed. For example, to calculate the whole DTW matrix for all cases it required more than 47 hours running non-stop. In the beginning of the ideation part the idea was to apply the algorithm to every single available joint, instead we opted to use only the

headset location sensor. Which brings us to the final limitation:
Use of only one joint: The idea of applying this algorithm to motion data was to use many points and detect not only the location of the headset, but the movement of every single joint. Being able to identify the most fine-grained motion of the hands and compare their execution. This however was not possible for two main reason, the first being the computational limitation, having to perform the same calculation for all joints (up to 53, 26 for each hand plus one for the headset location). This would increase the duration of the comparison up to 100 days of calculation. The second reason is that the other joints are not always available, since they are only read and recorded when the hands are visible, and dealing with such complexity is not manageable.

The set of steps performed greatly at identifying work practices, in our case 3 different ones. It is interesting to consider how the algorithm using 3 different metrics and taking into account all of them with different weights can yield such a clustering that can be interpreted both from a standpoint of motion data and similarity, in the case of trajectory comparison and from the temporal aspect of it and a process mining point of view. Especially helping the company to identify a problem in their data reading for PhotoCapture. One final discussion point regarding the weights, since the Silhouette method gave us as a result for better cluster with a very high weight for the Levensthein distance, meaning that it does not consider the other distances as much. However, the argument here is that Levensthein distance needs to be given more weight if we want an output that is relevant from a process mining point of view. Having more or less tasks compared to another cases need to penalize a lot more that just doing task in a slightly different time or with a different motion.

Future work in these fields is needed in order to apply different metrics of distance or use different joints in the equation, ensuring to capture not only the motion of the head but also how the different joints motion compare across different cases. For example, how different workers perform the task of lifting the lid, analysing their hand motion. Perhaps using a hierarchical weight for each joint section can provide us with more insight on which parts of the body are more involved in a task execution.

For example, in the healthcare field hand and finger joints motion can be given more weight than torso and legs, since the type of tasks performed is mostly using hands and head motion. Following the eyes' movement of an athlete during an intensive training can tell us more about which location the athlete focuses their attention on, this could be done by plugging the already implemented Quaternion DTW and tracking the eyes movement on a sphere instead of focusing on the movement.

Bibliography

[1] Applications of ar glasses in industry — augmented reality glasses, 04 2020.

[2] Dawon Kim and Yosoon Choi. Applications of smart glasses in applied sciences: A systematic review. *Applied Sciences*, 11(11), 2021.

[3] Christian Beecks, Marwan Hassani, Florian Obeloer, and Thomas Seidl. Efficient distance-based gestural pattern mining in spatiotemporal 3d motion capture databases. *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 11 2015.

[4] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[5] Optical motion capture dataset of selected techniques in beginner and advanced kyokushin karate athletes. *ResearchGate*, 2021.

[6] David Lamb, Joni Downs, and Steven Reader. Space-time hierarchical clustering for identifying clusters in spatiotemporal point data. *ISPRS International Journal of Geo-Information*, 9:85, 02 2020.

[7] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[8] Sönke Knoch, Shreeraman Ponpathirkoottam, Peter Fettke, and Peter Loos. Technology-enhanced process elicitation of worker activities in manufacturing. *Business Process Management Workshops*, pages 273–284, 2018.

[9] Maikel L. van Eck, Natalia Sidorova, and Wil M. P. van der Aalst. Enabling process mining on sensor data from smart products. *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*, 06 2016.

[10] Plotly Technologies Inc. Collaborative data science, 2015.

[11] Inc Motion Lab Systems. C3d.org - the biomechanics standard file format, 2021.

[12] Mokka - motion kinematic & kinetic analyzer, 2011.

[13] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *Levenshtein Distance: Information Theory, Computer Science, String (Computer Science), String Metric, Damerau?Levenshtein Distance, Spell Checker, Hamming Distance*. Alpha Press, 2009.

[14] Sönke Knoch, Shreeraman Ponpathirkoottam, Peter Fettke, and Peter Loos. Technology-enhanced process elicitation of worker activities in manufacturing. In Ernest Teniente and Matthias Weidlich, editors, *Business Process Management Workshops*, pages 273–284, Cham, 2018. Springer International Publishing.

[15] Jan Sedmidubsky and Pavel Zezula. Similarity search in 3d human motion data. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, ICMR '19, page 5–6, New York, NY, USA, 2019. Association for Computing Machinery.

[16] Andreas Aristidou, Daniel Cohen-Or, Jessica K Hodgins, Yiorgos Chrysanthou, and Ariel Shamir. Deep motifs and motion signatures. *ACM Transactions on Graphics (TOG)*, 37(6):1–13, 2018.

[17] Meinard Müller, Andreas Baak, and Hans-Peter Seidel. Efficient and robust annotation of motion capture data. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, page 17–26, New York, NY, USA, 2009. Association for Computing Machinery.

[18] Chuan Sun, Imran Junejo, and Hassan Foroosh. Motion retrieval using low-rank subspace decomposition of motion volume. *Comput. Graph. Forum*, 30:1953–1962, 09 2011.

[19] Liudmila Ulanova, Nurjahan Begum, and Eamonn Keogh. Scalable clustering of time series with u-shapelets.

[20] G. Atluri, M. Steinbach, K. O. Lim, A. MacDonald, and V. Kumar. Discovering groups of time series with similar behavior in multiple small intervals of time. In Mohammed Zaki, Zoran Obradovic, Pang Ning-Tan, Arindam Banerjee, Chandrika Kamath, and Srinivasan Parthasarathy,

editors, *SIAM International Conference on Data Mining 2014, SDM 2014*, SIAM International Conference on Data Mining 2014, SDM 2014, pages 1001–1009. Society for Industrial and Applied Mathematics Publications, 2014. Funding Information: This work was supported by NSF Grant IIS-1355072. Publisher Copyright: Copyright © SIAM.; 14th SIAM International Conference on Data Mining, SDM 2014 ; Conference date: 24-04-2014 Through 26-04-2014.

[21] J. Timmer, C. Gantert, G. Deuschl, and J. Honerkamp. Characteristics of hand tremor time series. *Biological Cybernetics*, 70:75–80, 11 1993.

[22] Jiawei Han, Jae Gil Lee, and Micheline Kamber. *An overview of clustering methods in geographic data analysis*, pages 149–188. CRC Press, January 2009. Publisher Copyright: © 2009 by Taylor & Francis Group, LLC.

[23] Jeremy Mennis and Diansheng Guo. Spatial data mining and geographic knowledge discovery—an introduction. *Computers, Environment and Urban Systems*, 33(6):403–408, 2009. Spatial Data Mining–Methods and Applications.

[24] Alan Dos, Santos Soares, and Antonio Apolinário. Real-time 3d gesture recognition using dynamic time warping and simplification methods.

[25] H. M. W. Verbeek, Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. Xes, xesame, and prom 6. In Pnina Soffer and Erik Proper, editors, *Information Systems Evolution - CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers*, volume 72 of *Lecture Notes in Business Information Processing*, pages 60–75. Springer, 2010.

[26] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh.

[27] C Daniel Meliza, Sara C. Keen, and Dustin R. Rubenstein. Pitch- and spectral-based dynamic time warping methods for comparing field recordings of harmonic avian vocalizations. *The Journal of the Acoustical Society of America*, 134(2):1407–1415, August 2013.

[28] Ian R. Cleasby, Ewan D. Wakefield, Barbara J. Morrissey, Thomas W. Bodey, Steven C. Votier, Stuart Bearhop, and Keith C. Hamer. Using time-series similarity measures to compare animal movement trajectories in ecology. *Behavioral Ecology and Sociobiology*, 73(11), November 2019.

[29] Bartosz Jablonski. Quaternion dynamic time warping. *IEEE Transactions on Signal Processing*, 60:1174–1183, 03 2012.

[30] François Petitjean, Jordi Inglada, and Pierre Gancarski. Satellite image time series analysis under time warping. *IEEE Transactions on Geoscience and Remote Sensing*, 50:3081–3095, 08 2012.

[31] dtaidistance.dtw_ndim — dtaidistance 2.2.1 documentation, 2021.

[32] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[33] Pratiwi Eka Puspita and Zulkarnain. A practical evaluation of dynamic time warping in financial time series clustering. In *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 61–68, 2020.

List of Figures

List of Tables