

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automatic marker labelling for MoCap systems

José Gomes

DISSERTAÇÃO



Mestrado em Engenharia Informática e Computação

Internal Supervisor: Miguel Velhote Correia Ph.D

External Supervisor: Vânia Guimarães MSc.

July 29, 2022

Automatic marker labelling for MoCap systems

José Gomes

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor Henrique Lopes Cardoso

External Examiner: Doctor Nuno Gonçalves

Supervisor: Doctor Miguel Velhote Correia

July 29, 2022

Abstract

Motion Capture is the process of recording the movement of a person or an object and converting such recordings into movement data to be used digitally for various purposes such as medical research, entertainment or military research. These types of systems generally use bright, white markers placed on the subject that reflect light back into the multiple cameras that film the action through different perspectives.

The main issue arises when it is time to attribute a meaning to what each individual marker represents. In the data extracted from the videos, all markers are, by themselves, meaningless, so there is a post-processing step called "labelling", which attributes to each marker a label (Right foot fifth metatarsal or Left Clavicle-Acromion Joint, for example). This labelling step is usually done automatically by Motion Capture software. However, due to occlusions and artefacts in the data, the need for manual labelling of the data is a common occurrence that stalls the workflow.

This dissertation documents the development and evaluation of an automatic labelling algorithm that is based on state of the art methods and explores different implementation options in more detail. The developed system can be used to label Motion Capture clips without requiring an initialization pose and no human input once the labelling has started by using structure and motion cues to model the movement of the captured subject

Labelling performances averaging 90% of correct marker labels were achieved, however the algorithm cannot be directly compared to other state of the art techniques due to lack of access to proprietary data sets and algorithms.

Keywords: Motion Capture, Labelling, Computer Graphics

Resumo

Motion Capture é o processo de gravar o movimento de uma pessoa ou objeto e converter essas gravações em dados de movimento para serem usados digitalmente para vários fins, tais como investigação médica, entretenimento ou investigação militar. Estes tipos de sistemas geralmente usam marcadores brancos e brilhantes colocados no sujeito que refletem a luz de volta para as várias câmaras que filmam a ação através de diferentes perspectivas.

A questão principal surge quando é hora de atribuir um significado ao que cada marcador individual representa. Nos dados extraídos dos vídeos, os marcadores não têm significado, por isso há uma etapa de pós-processamento chamada "labelling", que atribui a cada marcador uma etiqueta (Quinto meta tarso do pé direito ou Articulação clavícula-acromial esquerda, por exemplo). Esta etapa de etiquetagem é geralmente feita automaticamente pelo software de Motion Capture. No entanto, devido a oclusões e artefatos nos dados, a necessidade de rotulagem manual dos dados é uma ocorrência comum que perturba o workflow.

Esta dissertação documenta o desenvolvimento e avaliação de um algoritmo de etiquetagem automática baseado em métodos de última geração e explora mais detalhadamente diferentes opções de implementação. O sistema desenvolvido pode ser usado para etiquetar cliques de Motion Capture sem exigir uma pose de inicialização e nenhuma interação humana após o início da etiquetagem, utilizando informação sobre a estrutura e movimento para modelar o movimento do sujeito capturado.

O sistema alcança desempenhos de 90% de marcadores corretamente etiquetados, no entanto, o algoritmo não pode ser comparado diretamente com outras técnicas de última geração devido à falta de acesso a conjuntos de dados e algoritmos proprietários.

Acknowledgements

First, I want to express my gratitude to the Faculty of Engineering of the University of Porto and Fraunhofer Portugal for providing me with the tools and materials I needed to complete my degree. Second, I would like to express my deepest appreciation to my supervisors, Miguel Velhote Correia PhD. and Vânia Guimarães MSc. for the support and guidance that they provided me with during these months. Lastly, a very big thank you to my parents and to the group of peculiar people that get excited when it's 9 o'clock, for putting up with me and being my rubber duckies.

José Gomes

*The data used in this project was obtained from mocap.cs.cmu.edu.
The database was created with funding from NSF EIA-0196217.*

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	2
1.3	Document Structure	3
2	Background	5
2.1	Motion Capture	5
2.1.1	Magnetic Motion Capture	5
2.1.2	Mechanical Motion Capture	5
2.1.3	Optical Motion Capture	6
2.2	Labelling Concepts and Algorithms	7
2.2.1	Spatio-temporal consistency	9
2.2.2	Kalman Filter	9
2.2.3	Hungarian Method	10
2.2.4	Neural Networks	10
2.2.5	True-range Multilateration	11
3	State of the Art	13
3.1	Exploiting spatial and temporal consistency	13
3.2	Application of Machine Learning techniques	14
3.3	Yu’s method as a baseline	17
4	Methodology for labelling, testing and validating	19
4.1	The CMU Dataset	19
4.1.1	Marker set	19
4.1.2	Structure of a <i>.c3d</i> file	20
4.2	Initialization	21
4.2.1	Structure Model	22
4.2.2	Motion Model	23
4.2.3	Fitting the rigid bodies	24
4.3	Labelling	25
4.3.1	Getting candidate points for each marker	25
4.3.2	Calculating the best assignment	27
4.3.3	Filling gaps	28
4.3.4	Re-ordering the frame	28
4.3.5	Update Structure and Motion Models	28
4.3.6	Summary of the algorithm	29
4.4	Testing and Validating	29

4.4.1	Artificially destroying the data	29
4.4.2	Error Quantification	31
5	Results and Discussion	33
5.1	Labelling	33
5.1.1	Profiling	35
5.2	Rigid Body Fitting	35
5.3	Gap-filling	36
5.3.1	Resampling	38
5.4	Discussion	38
5.4.1	Fitting	39
5.4.2	Gap Filling	39
5.4.3	Performance	40
6	Conclusion and Future Work	41
6.1	Future Work	41
	References	43

List of Figures

2.1	(a) Magnetic MoCap Suit (b) Mechanical MoCap suit	6
2.2	Example of a Reflective Optical Motion Capture Setup	8
2.3	Simplification of the MoCap labelling pipeline	9
2.4	Example of spatio-temporal consistency	10
2.5	Visualization of a multilateration	12
4.1	Market set utilized by CMU's data set	20
4.2	Visualization of the structure model	22
4.3	Flow-chart of the algorithm.	26
4.4	Visualization of the candidate selection	26
4.5	Visualization of the index shuffling that occurs when labelling	29
5.1	Comparison of the performance between Kalman Filters and Unscented Kalman Filters and Nearest Neighbour.	34
5.2	Influence of the clip length (in frames) on the labelling time	34
5.3	Profiling of the full running system	35
5.4	Performance Comparison of Automatic vs Manual Rigid Bodies	36
5.5	Comparison of the performance between different gap sizes introduced periodically	36
5.6	Comparison of the performance between different gap types and sizes introduced manually	37
5.7	Analysis of the gap filling algorithm performance	37
5.8	Performance comparison between different resamplings	38

List of Tables

3.1	Table analysis of the different properties of the state-of-the-art approaches	16
-----	---	----

Abbreviations

MoCap	Motion Capture
DoF	Degree of Freedom
ML	Machine Learning
GMM	Gaussian Mixture Model
KF	Kalman Filter
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
RoM	Range of Motion
NN	Neural Network
CMU	Carnegie Mellon University
KD Tree	K-Dimensional Tree
fps	Frames per Second

Chapter 1

Introduction

This dissertation focuses on the process of labelling 3D marker points in a Motion Capture (MoCap) workflow. The main goal is to implement an algorithm that can correctly convert the raw spatio-temporal data recorded by the Motion Capture system into labelled markers that can be used to recreate, digitally, the originally recorded poses and movements. This algorithm needs to be robust to different poses and marker sets, as well as to imperfections in the captured data, such as occlusions of markers and extraneous ghost markers. The system should report back to the user the degree of uncertainty of the labelling process and help perform the possibly needed manual corrections.

1.1 Motivation

The use of MoCap techniques is widely spread around various industries. The fact that real-world motion can be recorded and used digitally has proved to be immensely useful across multiple scientific fields, application areas and businesses [Furniss, 1999, Rahul M., 2018]. It is to be noted that along this dissertation, the "Motion Capture" term will be referring to "Reflective Optical Motion Capture" unless specified – a more detailed explanation will be provided in [Section 2.1](#).

Because it is so ubiquitous, Motion Capture techniques have been the focus of many researchers for various reasons, such as improving the fidelity of the system, making the workflow and overall experience with the system easier and reducing the financial cost of acquiring such a system. This dissertation will focus on the workflow by making the labelling process as autonomous as possible without sacrificing the quality of the results.

The typical MoCap recording consists of dozens of markers per frame across multiples seconds of footage, generally recorded at more than 100Hz, which amounts to thousands of labels to fill per motion segment. This makes the labelling process very tedious and time-consuming, or even virtually impossible to accomplish by manual labour. While commercial systems do offer solutions for the labelling of the data, these solutions seldom yield results that require no user tweaking, making the user spend quite some time in these operations. Furthermore, labelling 3D markers is an operation that does not depend on artistic or technical knowledge possessed only by

the human mind. It is a problem with a single, correct solution determined by the physical nature of the object being captured.

These two aspects are on their own a perfect motivation for the automation of the labelling process, and combined create an even more solid argument for the need for automation. As such, removing the need for human input in this process can expedite the MoCap workflow and still allow for perfect results reliably.

The development and perfection of the MoCap workflow would be beneficial in all the areas where its use is applicable, such as:

- **Research:** Research areas such as biomechanical, medical and military research make use of MoCap to better comprehend and analyse motion [Pueo and Jimenez-Olmedo, 2017, Pachipala et al., 2018, Edwards, 2016].
- **Entertainment:** Movies and video-games have moved from handcrafting animations to capturing them with MoCap techniques to obtain more realistic results faster [Bregler, 2007, Radoff, 2008].
- **Education:** Although it is a more recent usage, there are many use cases for MoCap in education, where, similarly to research practices, it can help comprehend and visualize more complex concepts [Johnson-Glenberg and Megowan-Romanowicz, 2017].

1.2 Objectives

Quickly summarized, the goal of this dissertation is to produce a robust algorithm that maps labels to markers in the presence of occlusions and ghost markers. This goal can be broken down into various smaller objectives with different degrees of importance. The most important objectives that are expected to be accomplished by the end of this dissertation are considered to be the following:

- A labelling algorithm that is robust to imperfections in the data, such as ghost markers and occluded markers;
- A system that reports the degree of uncertainty associated with the labelling of each marker;
- A document that details the performance (efficacy and efficiency) of the developed system using existing datasets.

On a more secondary level, there are two aspects that will be taken into consideration, which are:

- The developed system should perform the labelling task in a reasonable amount of time – real-time execution is the target performance, but it is not the main focus;
- If the efficacy of the algorithm is not optimal in a certain use-case, the system should guide the user during the manual correction process.

The completion of all these objectives will result in a very well-performing system that can compete with the state of the art techniques that are mentioned in [Chapter 3](#).

1.3 Document Structure

Apart from this introductory chapter, this document contains five more parts: **Background, State of the Art, Methodology, Results and Discussion** and finally **Conclusion and Future Work**. [Chapter 2](#) contains the Background and aims to contextualize the problem and introduce some concepts that are required to understand the rest of the document. [Chapter 3](#) describes and compares the current state of the art methods that exist to solve the problem, highlighting their advantages and drawbacks. On [Chapter 4](#), the developed system is explained in detail, along with the used data sets and the methodology for testing and validating the algorithm. Next comes the evaluation results on [Chapter 5](#) that exposes data related to the algorithm's performance on various scenarios, trying to isolate the various components of the algorithm. Finally, in [Chapter 6](#) a brief summary of what was achieved (or not) is presented along with some thoughts about possible future work.

Chapter 2

Background

This chapter will focus on the knowledge necessary to better understand both the problem and the solution by introducing some concepts, giving an introduction to what is Motion Capture and explaining some common algorithms in detail.

2.1 Motion Capture

There exist many kinds of Motion Capture techniques and setups. Still, Motion Capture, in its essence, is the act of capturing the motion of a target – objects or people – and representing it digitally in some way.

2.1.1 Magnetic Motion Capture

Magnetic Motion Capture utilizes small magnetic sensors positioned on the body and connected to a host computer. Every sensor reports its position and rotation according to a known origin point [Kaiga et al., 2007]. This method has the advantage of being real-time and excluding the need for the labelling step since each sensor can be identified easily. However, since the sensors use magnetic fields, any metal objects will interfere with the recordings [Rahul M., 2018].

2.1.2 Mechanical Motion Capture

Another method denominated Mechanical Motion Capture consists of an exoskeleton containing sensors at each joint that measure the angle of that joint through time. The subject's body pose can be reconstructed through software because each sensor informs its relative position to every other sensor. Despite offering high precision recordings, this method has various drawbacks, such as limiting the subject's motion due to the wires and possibly heavy exoskeleton [Rahul M., 2018].

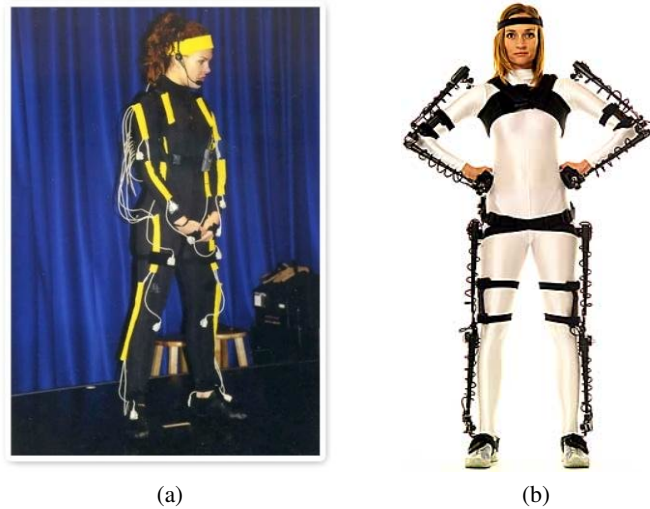


Figure 2.1: (a) Magnetic MoCap Suit (b) Mechanical MoCap suit

2.1.3 Optical Motion Capture

Currently, the technology that is acknowledged as the state-of-the-art technique is Optical MoCap. There are three main branches of Optical MoCap: **Markerless**, **Active** and **Reflective** Mocap. Although each method has its unique way of handling different aspects of MoCap, they all share one common point: the use of cameras. As the name indicates, Markerless Mocap does not make use of markers; however, Active and Reflective MoCap utilize some form of marker placed on the subject to aid the software in detecting the subject's body. In the methods that use markers, cameras positioned in different perspectives record the movement of a subject and, through algebraic calculations, the (x,y) coordinates of each camera allow the deduction of the (x,y,z) coordinates of the subject in the real world [Rahul M., 2018]. This 2D to 3D reconstruction can be thought of as the intersection of the rays that go from each camera's projection plane and pass through the 2D coordinate of the marker being triangulated. If the system utilized only one camera, the actual position of the marker would be hard to estimate since the point could be situated anywhere along the single projected ray. Having two or more cameras creates an intersection point among the various projected rays – if the cameras are perfectly calibrated, which is not a normal occurrence – and this point is, in fact, the real-world position of the marker [Guerra-Filho, 2005].

Optical MoCap introduced new problems in the MoCap pipeline, like occlusions – when no camera has a direct line of sight of a marker – and the need to infer marker rotation based on their relative position to each other – since there is no way to calculate their rotation based on the camera recordings alone.

Markerless Optical Motion Capture

Markerless Optical MoCap systems do not require any marker placed on the subject and are thus considered easier to set up and utilized in areas where marker placement can prove challenging, like sports [Nakano et al., 2020]. These methods utilize computer vision algorithms to identify human bodies and poses. These systems may also make use of RGB-D cameras to include depth information in the algorithms [Kim and Kim, 2014]. As mentioned and experimentally demonstrated by [Nakano et al., 2020], current Markerless MoCap techniques lack the accuracy and precision that can be achieved when using markers.

Active Optical Motion Capture

Active Optical MoCap systems require some sort of marker placed on the subject to aid in the three-dimensional reconstruction of the poses. Active markers are usually small LED's that, by flashing very rapidly, one at a time, allow for easy identification of what marker is active but include the need for wires that can be cumbersome and restrict the movement of the subject. This ease of identification is a vital aspect and an advantage of Active Optical MoCap since it removes the need for the step of labelling that will be described in Section 2.2.

Reflective Optical Motion Capture

Reflective Optical Motion Capture is a kind of Passive Optical MoCap in which the LED's and wires used in Active Optical MoCap are replaced by small reflective markers that bounce light back to the cameras. The reflective property of the markers facilitates their identification in the recorded footage, but this approach suffers from the occlusion problem mentioned in Subsection 2.1.3. The occluded markers may be due to interactions with external objects, the overlapping of the subject's limbs relative to the cameras' perspective, or even due to improper camera setups. There is also the possibility of extraneous markers appearing in the data. These markers, also called ghost markers, are caused mainly by shiny objects in the environment that are mistaken for markers by the capturing software or by improper camera calibration in terms of focus and/or aperture. While freeing the subject from cumbersome cables and exoskeletons, this approach comes with the added necessity of labelling, which will be described in Section 2.2. Figure 2.2 shows a Reflective Optical Motion Capture setup being used with multiple subjects.

There are a few more Motion Capture techniques, but the focus of this dissertation will be the Reflective Optical Motion Capture method – which will from now on be referred to as only Motion Capture or Mocap – and, more specifically, the labelling of markers.

2.2 Labelling Concepts and Algorithms

When dealing with passive markers, all markers are, by themselves, meaningless. The capturing software turns the physical markers, whose labels are attributed by their position on the body, into

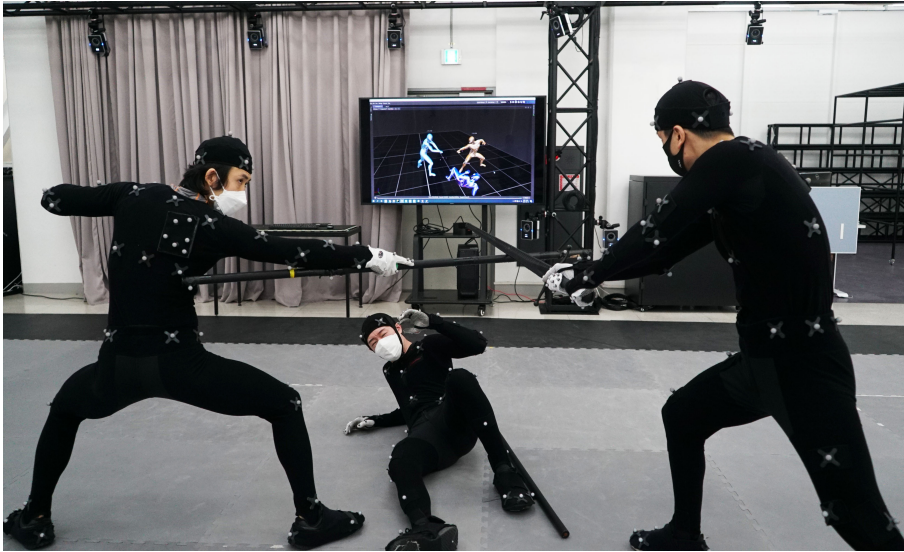


Figure 2.2: Example of a Reflective Optical Motion Capture Setup

a cloud of 3D points with no information about which marker is which. Because of this, a **labelling** step is required to match markers with their labels and restore the meaning of each marker, thus being able to reconstruct the subject's pose.

Correctly labelling the markers is a task that can be accomplished in a plethora of different ways. Modern MoCap software returns a labelled result, but the methods utilized are usually proprietary, and the result may require some manual corrections since the software may perform incorrect labellings in more complex areas of motion. Actual methods will be described in [Chapter 3](#).

The labelling problem has been described by many as an assignment problem. Ideally – when the captured data is perfect – having N physical markers yields N 3D points to be labelled. This means that a naïve brute force attempt could be experimented with. However, aside from being computationally impractical, the system still would need a verification method to confirm the veracity of each brute force attempt. This veracity verification is sometimes achieved by manually labelling one frame of motion and later ensuring the consistency of the trajectories and thus the labellings of each marker. While this approach yields good results, it fails in the presence of occlusions and ghost markers because matching a frame's markers to the next ones stops being an N to N assignment problem and becomes a much more complex obstacle. If a marker is missing, its position needs to be estimated to maintain the smoothness – or abruptness – of the captured movement. On the other side, if there are more markers than expected, it must mean that some markers are not supposed to be in the data and are actually imperfections in the captured data that need to be ignored. A more obscure phenomenon could occur where, in the same frame, a marker gets occluded, and a ghost marker appears in the data. This, in practice, would still yield an N to N assignment problem, but in reality, it would be a case where the system needs to discard the ghost marker and estimate the position of the real missing marker. The three described phenomenons

make the automatic labelling of MoCap data a complex problem that has become the focus of study of many researchers. A very simplified diagram of the Motion Capture pipeline is presented in [Figure 2.3](#)



Figure 2.3: Simplification of the MoCap labelling pipeline

Given the amount of existing research regarding Labelling in MoCap, it is only natural that some approaches have some aspects in common. Mainly due to the fact that all the methods try to solve the same problem, some algorithms and techniques have become ubiquitous in the field because they have been proved to work with excellence before. Some of these algorithms and concepts will be briefly described in this section.

2.2.1 Spatio-temporal consistency

As the name implies, spatio-temporal has to do with data that was recorded across both space and time. Spatio-temporal consistency derives from the fact that some data – as is the case of a MoCap recording – has some kind of relation between the spatial and temporal domain. [Figure 2.4](#) helps visualize how spatio-temporal consistency is utilized in MoCap. If the positions, velocities and accelerations of the green and the blue markers at frame 1 are known, their expected position at frame 2 can be calculated. Not only that, but in the case of MoCap, many markers have stable distance relationships – markers placed on the same limb of a subject are not likely to move relative to each other –, meaning that a marker position can be inferred if some other related marker's position is known due to the somewhat constant distance between them over time – in the example this can be thought of as estimating the blue marker position in frame 2 based only on the green marker position, in frame 2, and their known relationship in frame 1.

2.2.2 Kalman Filter

Rudolf E. Kalman first described the Kalman Filter in 1960 [[Kalman, 1960](#)]. The filter produces an estimate of the state of a system – past, current or future – based on information that has been recursively measured, possibly with noise. These measurements are filtered, and the hidden state variables of the system can be deduced from them. This type of filter is very useful for linear applications due to its internal use of matrices and linear algebra. However, most real world applications are not linear, so the Extended Kalman Filter (EKF) and later the Unscented Kalman Filter (UKF) were developed.

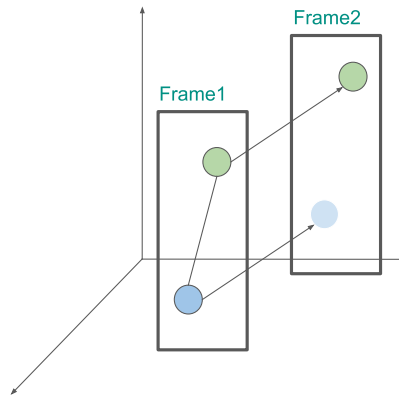


Figure 2.4: Example of spatio-temporal consistency

As stated by [Julier and Uhlmann, 1997], an EKF transforms a non-linear system into a linear system through complex mathematical operations. This makes it difficult to implement and use. On the other hand, the UKF, developed by Julier and Uhlmann, is a much easier to implement filter that by using a set of points to parametrize the mean and covariance (called Sigma Points) can generalize the problem for non-linear systems.

These filters are extensively used in navigation systems, target tracking applications and various other problems [Kim and Bang, 2018], even when the specific nature of the system is not well-defined¹. For MoCap applications, Kalman Filters (and all their derivatives) are generally used to estimate the position of a marker in the current frame, but the actual purpose of this estimation may vary. Since both the regular Kalman Filter and the UKF perform the same operations, the use of "Kalman Filters" throughout this document can mean any of the two, unless specified.

2.2.3 Hungarian Method

The labelling of MoCap markers can be considered as an assignment problem, where N labels need to be assigned to N markers. The Hungarian Algorithm is used to find the minimum cost in N to N assignment problems like the labelling problem. This algorithm was proposed by [Kuhn, 1955] and consists of an iterative approach that reaches an optimal solution in $O(N^3)$.

2.2.4 Neural Networks

In the 1960s, scientists started trying to have a computer mimic the human brain and make it solve complex problems. This technique proved to be quite successful, allowing machines to perform sophisticated tasks such as pattern recognition and the ability to learn. Neural Networks (NN), as these systems are called, are composed, like the human brain, of connected neurons in many ways. These digital neurons receive input data from other nodes, weigh them – give more weight

¹A more in-depth but still easy to understand explanation is available at kalmanfilter.net and github.com/r1abbe/Kalman-and-Bayesian-Filters-in-Python/

to more important nodes – and produce an output according to some **activation function**. But how do these networks of nodes produce any sensible result? The answer lies in training/transforming the weights of each node's inputs until the outputs achieve the desired outcome. NN's require a training phase where data is passed through the network, and the output is compared to the expected result. The sum of the differences between these two values across all training examples is called **total error**. If the total error is zero, then the Neural Network is perfect (meaning that it outputs the correct values for all training examples) and the closer this error is to zero, the better is the network. The weights are updated in order to minimize the error, but the specifics of this process are beyond the scope of this dissertation.² After enough training is performed, the trained model can be applied to data of the same nature and the network is expected to output results according to the information it was trained with [Krogh, 2008].

Neural networks have a huge advantage when working with more complex problems where defining a mathematical model is difficult or even close to impossible. Training a NN removes the need for such a model and instead lets the computer figure out the relationships inside the data. However, Neural Networks usually require some kind of preprocessing of the data in order to comply with the format of the input nodes, and the training times may be an essential factor that holds back the development.

2.2.5 True-range Multilateration

True-range Multilateration is a method of finding the location of a target object based on the location of various stationary objects. In Figure 2.5, the location of point T was found by utilizing the locations of points S1, S2 and S3 and some kind of search radius defined graphically by the coloured circles. In a motion capture context, this can be used to infer the position of a marker if there are known distance relationships between it and other markers. This algorithm will come handy in Subsection 4.3.3

²An excellent in-depth explanation about Neural Networks can be found in neuralnetworksanddeeplearning.com

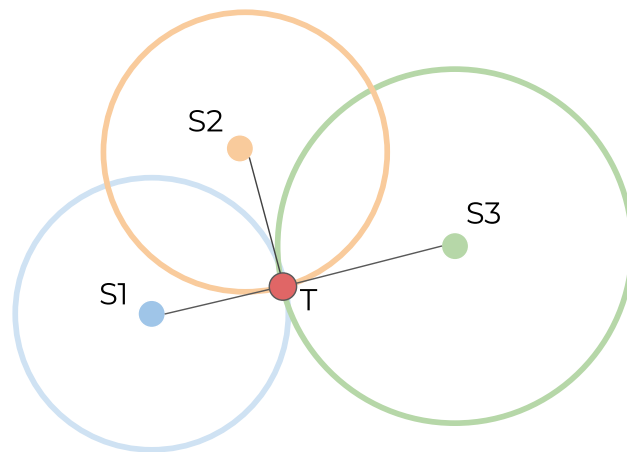


Figure 2.5: Visualization of a multilateration

Chapter 3

State of the Art

This chapter will delve into the state of the art of automatic labelling systems, pointing out what makes different approaches unique, categorizing them and comparing their performances. The reviewed solutions will be summarized in a table to better understand the relationship between chosen methods, their advantages, their trade-offs as well as their performance.

A broad categorization can be made by separating the methods in those which use "hand-crafted" mathematical models – through probabilities, geometry and biomechanical constraints – to exploit the spatial and temporal consistency of the problem and those which utilize machine learning techniques to let the software figure out some aspect of the problem.

3.1 Exploiting spatial and temporal consistency

The work of [Yu et al., 2007], proposes the use of the spatio-temporal consistency of the movement as well as the inter-marker relationships in order to create a consistent tracking algorithm. This method uses the first 50 frames of motion to calculate a Structure Model and a Motion Model (these models continue to be updated throughout the labelling process). The Structure Model is composed of a Distance Matrix that represents the distance between each pair of markers and a Standard Deviation Matrix that defines the standard deviation of the distances between each pair of markers. Markers whose standard deviations are low enough are grouped to form rigid bodies, inferring that the markers belong to the same limb and their distance to each other should not vary with time. The Motion Model is calculated by using Kalman Filters [Kalman, 1960] for each marker – creating a means of predicting the motion of a marker given its previous position and movement properties. During the labelling process, the algorithm exhaustively searches for the best matching markers of each rigid body by means of a Score Function that makes use of the Structure Model. The rigid bodies that fail to be matched – due to occlusions or ghost markers – are fitted with the information provided by the Motion Model. This approach was shown to work even with multiple targets without manual intervention.

Another way to take advantage of the spatio-temporal consistency of the movement was presented by [Meyer et al., 2014]. This approach fits a skeleton into the cloud of 3D points and

requires a T-Pose initialization process to get the initial labels of each marker. After the initial labelling, the spatio-temporal consistency of the movement is used to label subsequent frames utilizing the Hungarian method with a cost function related to the spatial distance between markers from one frame to another. If labelling by temporal consistency is not possible due to missing markers, the skeleton model is used to estimate where the missing marker would be positioned – keeping in mind the limited DoF of some joints like the knee. This approach is unique because it requires no user intervention but introduces the need for a T-Pose stage in the captured data.

The team of [Alexanderson et al., 2017] devised a method for MoCap in more flexible body parts such as the hands and the face, where inter-marker relationships have lesser meaning. Capturing the hands is especially prone to long occlusions, leading to deterioration in the temporal cohesion, making it disadvantageous. With this in mind, the algorithm bases itself on the spatial domain instead of the temporal domain. The system uses Gaussian Mixture Models [Reynolds, 2009] to model the position of the markers and Kalman Filters to predict their movement. The algorithm maintains its high performance by making N possible marker assignments and, when uncertain, postponing the decision until more evidence is available. This technique has the advantage of easily recovering from more prolonged occlusions because it works on the spatial domain but requires a short training section to analyse the full Range of Motion of the markers to initialize the GMM's.

In another research, [Xia et al., 2017] suggested a solution involving Graph Matching by thinking about the problem as a correspondence problem. In this solution, a graph is created in which nodes represent markers and edges represent a relationship between nodes in the same limb, meaning that the distance between them is fixed over time – the edge length will not change. Every frame, the matching cost of a labelled node at the previous frame with an unlabelled node at the current frame is calculated – only nodes that fall within a certain radius calculated by a Kalman filter that predicts the movement of a node are considered for each marker – and the best matches for every node are found by applying the Hungarian algorithm. Missing markers are recovered from the available graph data by reconstructing the pose through Inverse Kinematics. This method works with various subjects simultaneously, but the performance is highly affected by the increase of markers. It also requires a T-Pose initialization phase to set up all the nodes and edges.

3.2 Application of Machine Learning techniques

Delving into more machine learning related approaches, [Jiménez Bascones et al., 2019] introduced an AdaBoost algorithm that utilizes several weak classifiers and finds the appropriate classifiers for the motion in training. The weak classifiers are defined over geometrical features like the distance between two points and the similarity ratio between two segments. By applying these classifiers over the cloud of points, a decently good classifier may be trained by combining the minimum amount of weak classifiers that correctly satisfy the labelling problem. Training this AdaBoost technique is a process that can take several hours on a conventional laptop computer, and requires training for every new type of motion that needs to be labelled.

Thinking about the combinatorial, or more specifically permutational nature of the problem, [Ghorbani et al., 2019] proposed a Permutation Learning approach in which a feed-forward deep residual neural network was trained to recover the correct order of the original markers from a shuffled array of markers. This machine learning approach, once trained, can only be used with recordings with the same marker layout.

In recent years, researchers have been trying to apply Deep Neural Networks to every possible problem, and Motion Capture is no exception. In 2021, [Clouthier et al., 2021] proposed a Neural Network Architecture that outputs label assignment probabilities based on marker trajectories (position, velocity and acceleration). Utilizing Neural Networks requires pre-processing the data in a way that can be input into a Neural Network. In this case, data was split into windows of 120 frames for each visible marker, making use of the marker trajectories that can be extracted by MoCap software. If there are gaps smaller than 12 frames in a trajectory, they are linearly interpolated. Later, to handle ghost markers, for each 120 frames window, if there are more markers than labels, the markers that are present for the most time during the window are retained, or if that metric is ambiguous, the markers closest to the marker of interest in that window are kept and the others discarded. The position, velocity and acceleration of each marker are then calculated and a matrix of size $WindowSize \times (5 * (NrLabels - 1))$ is created for input to the NN. The output of each NN is an $1 \times NrLabels$ vector for each window that is input into the NN. This vector corresponds to the probabilities of each label being the correct one for that marker and window. The real assignments are then chosen based on these probabilities via the Hungarian Method. To avoid cases where the same label is attributed to multiple markers, the assignments are weighted and only the label with the highest probability is assigned. This method's performance is limited to the training data provided to it and to combat this Clouthier et al. introduced a transfer learning module that allows the amplification of the knowledge of the NN by adding more training data after the initial training. Another drawback of this method is that it does not fill gaps bigger than 12 frames, thus leaving the motion data incomplete.

It is to note that all the mentioned algorithms can work in real-time with excellent labelling accuracy and precision in the problems that they were tailored for – upwards of 95% of correctly labelled markers in varying marker sets and degrees of occlusion.

Author	Main Method	Initialization	Advantages	Drawbacks	Marker sets
Yu et al. 2007	Rigid body fitting	Trains Structure and Motion models as it labels	Works with multiple subjects	Requires many markers per rigid body	Multiple interacting targets
Meyer et al. 2014	Skeleton Tracking	T-Pose	Zero user intervention	Requires T-Pose	Single subject full body
Alexanderson et al. 2017	GMM's, Kalman Filters, Viterbi Algorithm	Short training with full RoM and Initial Labelling	Adapts well to fingers and face MoCap	Requires post-process gap-filling methods	Hand and face
Xia et al. 2017	Graph Matching	T-Pose	Works with multiple subjects	Requires T-Pose and can be slow with multiple subjects	Multi subject full body
J. L. Jiménez Bascones 2019	Adaboost	Training Several weak classifiers and the Adaboost model	Not reliant on temporal coherence	Not robust to new datasets	Single subject full body
Ghorbani et al. 2019	Permutation Learning	Training of a NN	Zero user intervention	Same marker set as training data	Single subject full body
Clouthier et al. 2021	Deep Neural Network	Training of a NN	Transfer Learning capabilities	Somewhat limited to training data	Real and simulated full body single subject

Table 3.1: Table analysis of the different properties of the state-of-the-art approaches

With the analysis present in [Table 3.1](#), a few key decisions can be made to rule out some methods that do not meet the desired requirements. The two machine learning techniques [[Jiménez Bascones et al., 2019](#)], [[Ghorbani et al., 2019](#)] and [[Clouthier et al., 2021](#)] have the significant drawback of being tied to their training dataset, thus not being robust to different types of movement (even with the Transfer Learning technique in [[Clouthier et al., 2021](#)])). Having to manually label a training dataset for every type of motion and marker configuration is a time-consuming and laborious job that can be avoided with other techniques.

By excluding the Machine Learning techniques, the techniques that explore the spatio-temporal consistency of the movement become the go-to solution. Nevertheless, some of them present benefits that make them more suitable for what this dissertation aims to develop (or drawbacks that make them undesirable). For example, [[Meyer et al., 2014](#)] and [[Xia et al., 2017](#)] depend on the data starting with the subject in a T-Pose. This requires extra steps when recording the MoCap clips and puts a constraint on the number of existing datasets that the methods can label. Existing clips in which the subject does not start in a T-Pose cannot be labelled by these approaches.

Despite being a very interesting technique, the method presented in [[Alexanderson et al., 2017](#)] will not be studied because it is tailored only for hand and face motion capture.

3.3 Yu's method as a baseline

At its core, the approach presented by [[Yu et al., 2007](#)] is more of a tracking algorithm than a labelling one. The method does not care about the actual labels of the markers, but rather the correspondence of markers between frames and the consistency of their relationship with each other. This allows for a very versatile system that has a few major benefits:

- It does not require any special initialization pose. Initialization poses are normally used to get an initial label of every marker in order to maintain it throughout the labelling process. Since this method does not care about the labels, these poses are unnecessary.
- It is robust to any type of marker set, as long as each limb of the subject is covered with multiple markers (which is a common occurrence if good MoCap practices are followed). Missing marker positions are inferred based on the known positions of other markers in the same rigid body, thus every rigid body needs various markers in order to be tracked properly.
- It can track virtually any type of movement without any previous knowledge of the motion type. The tracking nature of the algorithm makes it oblivious to the type of motion being performed.
- It is, theoretically, able to track multiple interacting targets since the movement of markers is only constrained by their relationship inside each rigid body of each subject. This property of the method will not be verified due to the lack of datasets containing multiple interacting targets.

One minor drawback of the method is that, since it just tracks the markers, it has no knowledge of the actual labels. This requires the user to know the correct labels of one frame of the clip in order to get the actual labels, but this can be any frame of the clip after running the system due to the nature of the *.c3d* file system (more on *.c3d* files in [Subsection 4.1.2](#)). This knowledge of the correct labelling can be provided by the MoCap software that processed the recordings, or be manually defined by the user.

Despite being the oldest of all the methods studied, this is a very versatile system that can (supposedly) be inputted with any type of motion data, regardless of marker set, movement type and even the amount of subjects and their interactions. For these reasons, Yu's method will be used as a baseline for the implemented system.

Yu's method is very well explained in its original publication, starting at the training phase and progressing through the whole algorithm until the labelling is finished. Every major step of the algorithm has its purpose defined and fits the in general explanation that the publication provides. Despite this fact, certain unthought of aspects reveal themselves while trying to create an algorithm that accomplishes the same tasks. This dissertation will describe in detail the implemented system and reference, when necessary, Yu's method, which it very closely resembles.

Chapter 4

Methodology for labelling, testing and validating

This chapter will start by describing the dataset used, highlighting its relevance for the project. Then, a detailed description of the implemented system will be presented, along with justifications for the decisions taken and explaining the relationship between each stage and its necessity. Finally, the metrics used to evaluate and validate the results will be introduced.

4.1 The CMU Dataset

In order to implement, test and validate the developed algorithm, some form of real data was needed. There are a lot of motion capture files circulating around the internet, but the chosen dataset was provided by Carnegie Mellon University's (CMU) Graphics Lab. This dataset is available at mocap.cs.cmu.edu and was chosen for the quality of its recordings as well as the variety of movements and subjects.

The dataset is divided into various subjects and movement categories, with more than one hundred subjects and thousands of clips to use. This dissertation focused on subject 143 due to its general types of movements. The clips used include walking, running, jumping, waving, kicking, etc. – with a total of 26 different clips. The subject 94, focused mainly on dance clips, was also used, specifically the motion number 7 because of its erratic movement with fast changes in acceleration of almost every marker. All the clips used were recorded at 120fps.

4.1.1 Marker set

Regarding the marker set, there can be many possible arrangements and the Motion Capture Systems manufacturers usually recommend a certain marker configuration with clear instructions about the positioning of each marker – for example Vicon's Reference Guide available at <https://docs.vicon.com/display/Nexus213> (accessed on June 22nd, 2022). There are even studies that try to find the optimal marker placement for specific types of MoCap like [Reverdy et al., 2015] and [Malus et al., 2021].

The marker set utilized by the CMU dataset is shown in [Figure 4.1](#) and it follows closely to the Vicon 512 manual. A more in-depth description of the procedures taken to place the markers can be found at <http://mocap.cs.cmu.edu/markerPlacementGuide.pdf> (last accessed on June 22nd, 2022). The marker set contains 41 markers, placed in strategic places such that every limb of the subject has at least 3 markers attached. There is also special care to create asymmetry in the markers by placing some markers of the right limbs at a higher position than their corresponding left marker (for example, the "LTHI" marker is closer to the knee than the "RTHI" marker). This asymmetry allows for an easier distinction of the left and right side of the subject.

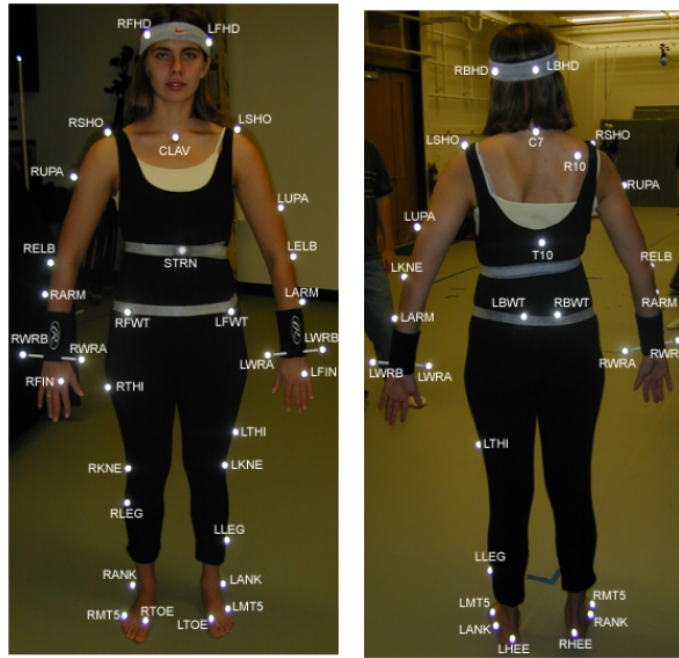


Figure 4.1: Market set utilized by CMU's data set

4.1.2 Structure of a *.c3d* file

Even though there are a lot of file formats used in the MoCap world¹ and the CMU dataset provides various file formats, this dissertation uses the *.c3d* file format. This file format was chosen because it contains the raw 3d points that were detected by the capture system, unlike file formats like the *ASF/AMC*, which contain a more processed version of the movement already with a skeleton and other movement information.

Every *.c3d* contains three important pieces of information:

- The **frame rate** at which the clip was captured (usually 120Hz). This information is useful for performing time calculations between frames.

¹A list of files can be seen in https://en.wikipedia.org/wiki/List_of_motion_and_gesture_file_formats

- An **ordered list of labels** that corresponds to what should be the real order of the points at each frame.
- A **list of frames**, each containing a list of 3D points. In an unlabelled clip, the points inside each frame have an arbitrary order. When labelled, the points of every frame x should be in the same order of the frame $x + 1$. This order corresponds to the order of the label list. For example, if the first entry of the label list is 'RHEE', then, in a labelled clip, the first entry of every frame should correspond to the position of the 'RHEE' marker.

4.2 Initialization

The method starts by "training" a motion and structure model [Yu et al., 2007]. The term "training" in this context does not mean the same thing it usually does in an ML context, for example. The training performed here is more of an initialization step that gathers information of a few frames in order to set the algorithm in the right direction. This training is done in the first frames of each clip, and there are some assumptions:

1. The training frames contain enough movement to figure out the relationships between markers. If the subjects remain static in the training frames, the whole body might get grouped as one single rigid body, so a minimal amount of movement is required.
2. If the clip contains recordings of multiple subjects, there are no interactions between them in the training frames, meaning that the subjects do not touch each other. Interactions between subjects would create relationships between markers of different subjects that are not real.
3. The frame-to-frame association can be done via Nearest Neighbour matching (points in frame x are associated to their closest match in frame $x + 1$). In this stage, the algorithm has very little information regarding the movement, so this technique is used in a small amount of frames (approximately 0.41 seconds).
4. The training frames contain no gaps. For the Nearest Neighbour algorithm to work effectively, every marker in frame x must be associated with one marker in frame $x + 1$, and no two markers can have the same association (the pairing of markers between frames must be a one-to-one relationship).

The number of frame to use for training can be adapted to best fit each clip and to better suit different capture rates. For example, Yu's method utilizes the first 50 frames of clips captured at 120fps, which amounts to approximately 0.41 seconds of motion. The training consists of, for each frame in the training range, calculating the Nearest Neighbour associations between its markers and the previous frame markers and feeding those points to the structure model and motion model that will be discussed next.

4.2.1 Structure Model

In theory, the structure model is made up of two symmetric matrixes **A** and **D**. **Matrix D** is a distance matrix that contains the average distance between each pair of markers. **Matrix A** is the standard deviation of the distances between each marker pair during the clip [Yu et al., 2007]. These two matrices help create a "skeleton" for the subject being recorded. Markers that maintain a somewhat constant distances to each other during some frames should be able to be grouped into a rigid body (likely a body segment of the subject, such as an arm or a leg). This model is later used to calculate the best fit of points for a rigid body's markers through a score function (see Subsection 4.3.2). It is also used to infer a point position in possible case of the marker's tracking being unreliable (see Subsection 4.3.3).

In practice, these matrixes are stored in a decomposed form to allow for the iterative calculation of the average distance and standard deviation. This decomposition has three components: an accumulation matrix of the distances *Sum*, an accumulation matrix of the square of the distances *Square* and a frame count *n* that refers to the amount of frames that were accumulated in the previous matrices. These values can be used to calculate the Average and Standard Deviation of the distance, according to the following formulas:

$$D_n(i, j) = \frac{Sum(i, j)}{n} \quad (4.1a)$$

$$A_n(i, j) = \sqrt{\frac{Square(i, j)}{n} - \left(\frac{Sum(i, j)}{n}\right)^2} \quad (4.1b)$$

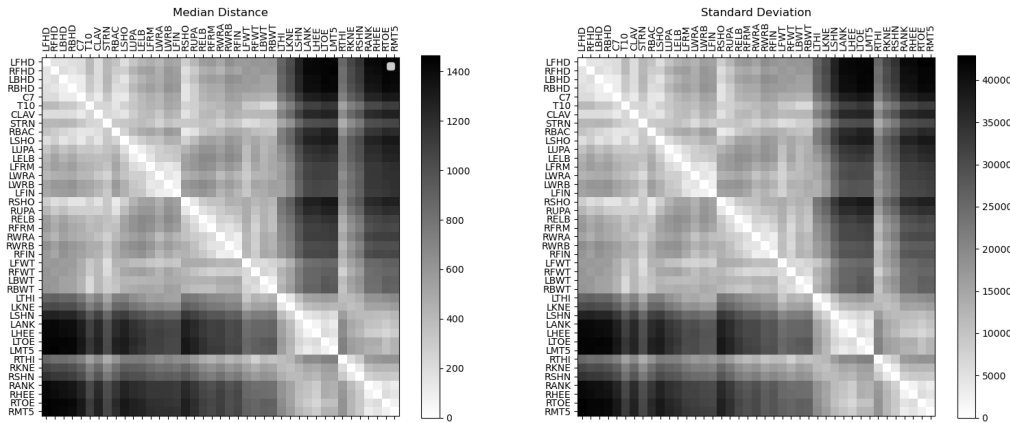


Figure 4.2: Visualization of the structure model in a labelled clip. Whiter values represent lower values (in millimetres). Note the white diagonal that represents the distance of a marker to itself - zero.

Equation 4.1a and Equation 4.1b are used only when necessary throughout the algorithm and allow for on-the-fly calculation of **Matrix A** and **Matrix D** for every pair (i, j) of markers. In Figure 4.2, the two matrices that compose the structure model can be visualized.

When the nearest neighbour correspondences are established (the marker positions at frame $x+1$ are known), a new distance matrix can be calculated and added to the *Sum* matrix. Likewise, a new matrix containing the square of the distances can be calculated and added to the *Square* matrix. Finally, after incrementing the frame count n by 1, the equations 4.1a and 4.1b are set up to compute matrix D and A at frame n

4.2.2 Motion Model

The motion model is made up of a Kalman Filter for every marker [Yu et al., 2007]. As was explained Subsection 2.2.2, a Kalman filter is a method that produces an estimate of one or more variables, based on previous measured information of the same or other related variables. The system allows for two types of Kalman Filter to be used: the basic Kalman Filter and the Unscented Kalman Filter. The system uses the Python library FilterPy (<https://github.com/rlabbe/filterpy>, accessed on June 18th, 2022) both for the implementation of the regular Kalman Filter and the Unscented Kalman Filter.

The state vector s , defined in Equation 4.2a, has 9 variables. The first three variables correspond to the x, y and z position of the marker; the following 3, v_x, v_y, v_z represent the marker instant velocity calculated by the difference in position from one frame to another; and finally, the last three variables are the acceleration of the marker, calculated by the change in velocity between the last three frames. Each filter's State Transition Function F , Measurement Function H and Measurement Noise Function R is initiated according to Equation 4.2b and Equation 4.2c. This formulation was based on [Gomes et al., 2021]. In Equation 4.2b, dt is the delta time between two frames. I_9 is the identity matrix of size 9x9 and the value 0.5 is the error factor that is associated with the motion capture system.

$$s = \begin{bmatrix} x, y, z, v_x, v_y, v_z, a_x, a_y, a_z \end{bmatrix}^T \quad (4.2a)$$

$$F = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} & 0 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 & \frac{dt^2}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2b)$$

$$H = I_9 * s \quad R = 0.5^2 * I_9 \quad (4.2c)$$

In the case of the Unscented Kalman Filter, the appropriate conversion between matrices and

functions is performed, and the Sigma Points were generated following [Merwe et al., 2004] formulation that parametrizes the points using an alpha (α), beta (β) and kappa (κ) value. This way and continuing to follow [Gomes et al., 2021], the values were set to be $\alpha = 0.1$, $\beta = 2.0$ and $\kappa = 0.0$. The α determines the spread of the points and should be a positive, small value (0.01 to 1.0). The β value incorporates prior knowledge of the distribution of the mean, and a value of 2 is optimal for Gaussian distributions. Lastly, κ is a secondary scaling parameter, usually set to 0 or to $3-n$, with n being the number of variables in the state vector s .

Each frame, the motion model is updated with new measurements taken, as explained before, from the previous three frames. The motion model is expected to provide a good tracking of the markers in the data and a good estimation of the position of each marker in the next frame.

4.2.3 Fitting the rigid bodies

Once the training is complete and the structure model is defined, the markers can be grouped into rigid bodies based on some criteria. This grouping serves mainly to create tight relationships between markers. The ideal rigid body only contains markers that maintain the same distance to each other during all the clip. For example, all the markers of the head should remain stationary relative to each other, since the head (more so than any other part of the body) is a very rigid structure. During the labelling phase, the rigid body relationships will be used to calculate how well of a match a point is to a marker given the rest of the relationships inside a rigid body. These relationships will also be used to possibly infer the position of missing markers, since their relative position to each other is known (to some extent).

There are many possible methods to group markers into rigid bodies, but naturally, the information gathered by the structure model is useful for this purpose, particularly the Standard Deviation matrix. Markers that maintain a low standard deviation between each other can be assumed to belong to the same rigid body.

The implemented system uses a simple approach of getting for each marker M the list of n markers with the lowest standard deviation. Usually, the value of n in a MoCap marker set is between 3 and 5. This method, has the disadvantage of forcing every rigid body to have a fixed amount of markers. This makes it so that some rigid bodies contain markers that don't fit in as well as the other, making the "rigid" body less "rigid". In these extra markers, standard deviation usually rises as the clip progresses. This approach also allows a marker to belong to various rigid bodies. This redundancy eliminates single points of failure in the case that a rigid body is badly fit and is not able to produce valuable structure information – another rigid body containing the same marker might be able to provide more accurate indications of the rigidity.

Because of this disadvantage, the system also allows manual input of which markers belong to the same rigid body. This allows for more accurate results at the expense of some manual work for each different marker set used.

4.3 Labelling

Until this section, the algorithm prepared the **Structure** and **Motion Models** for the labelling process utilizing some naïve methods just for initialization's sake. Now, the algorithm starts the actual labelling that can be resumed to these steps:

- Retrieving the candidate matches for each marker
- Calculating the rigid body that is best suited with the current candidates, until every rigid body **that can be satisfied** is satisfied
- Filling the gaps left by the rigid bodies that could **not** be satisfied
- Reordering the frame and update the **Structure** and **Motion Models**

Enumerating the steps like this may seem confusing at first but separating the algorithm into distinct phases helps to understand the different parts of the algorithm, their necessity and how they relate to the next phase. Each step will be further explained in the following sections. In order to understand the algorithm, it is crucial to keep in mind the models that were trained before:

- Structure Model - represents how each marker relates to any other marker with measures of average distance and standard deviation
- Motion Model - tracks each individual marker and can estimate a marker position at the next frame

and the rigid bodies that divide the markers into groups of markers whose movement is closely related in the subjects body. [Figure 4.3](#) shows the flow of the algorithm. At this stage, the Motion and Structure Model have been created, and the labelling algorithm is ready to start. At the end of labelling each frame, new information is provided to the Motion and Structure Model to keep them updated.

4.3.1 Getting candidate points for each marker

The labelling of each frame starts by getting candidate points for every marker [Yu et al., 2007]. The developed method represents the frame points as a k-dimensional tree (KD Tree) in order to facilitate the spatial search. Very simply put, a KD Tree is a data structure that organizes data in k-dimensional space (3-dimensional space, in this case). The KD Trees are useful for various kinds of operations in 3 dimensional data, but they will be used in this case for finding the candidates for a marker that lie inside some defined range.

The state estimation properties of the motion model are used to figure out a general area where the markers should be in the next frame, and those positions are used as the search points for candidates. After that, the algorithm searches for points in the current frame that are in a radius r of the selected search points, and the points that are found are selected as candidates.

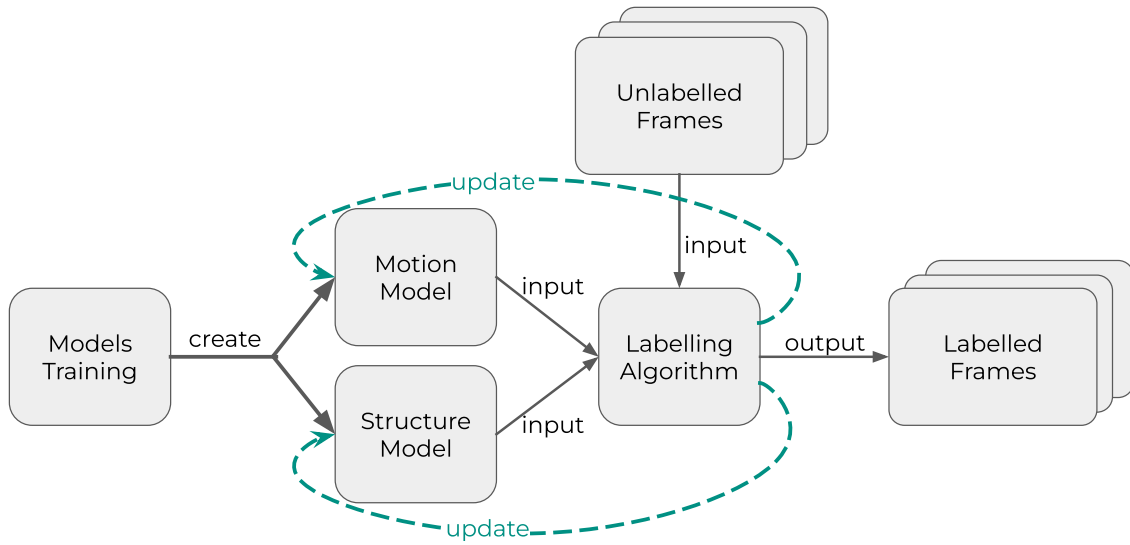


Figure 4.3: Flow-chart of the algorithm. Very similar to [Yu et al., 2007]

Looking at the example present in Figure 4.4, the green and blue points with dark outline are the search points that were retrieved with the Kalman Filters (motion model); the blue and green points without outlines are the current frame points that will become candidates; and r_1 and r_2 are the corresponding search radiuses for the green and blue points. A distinction is made for r_1

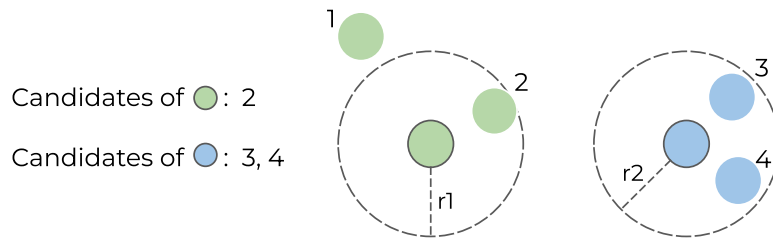


Figure 4.4: Visualization of the candidate selection

and r_2 in the example because there is no reason for the search radius to be a fixed value. The velocity information present in the motion model is a good indicator of how much the point has moved since the last frame, and the search area can be increased or decreased proportionately to the velocity, thus possibly reducing the search area for performance and increasing it for better candidate assignment if necessary.

The implemented system allows for essentially two search procedures: "fixed" and "velocity dependent". The "fixed" approach, as the name implies, utilizes a constant, user provided search radius for every marker. The "velocity dependent" method is a little more nuanced, and allows the search method to be dependent on a percentage of the speed of the marker and clamped to a minimum and maximum value. A percentage of the speed is used because the units are in meters/second and from one frame to another the time difference is very inferior to one second.

As for the clamping, the minimum and maximum were introduced to combat markers that estimate a very low or very high velocity, creating search areas that include no markers or too many markers respectively. All these values can be adjusted by the user to better suit the MoCap clip. Setting the system to have higher search radiuses makes the system more accurate since more assignment options are explored, but also significantly slows down the labelling process because of the same reason – more options are explored, as explained in [Subsection 4.3.2](#).

4.3.2 Calculating the best assignment

Once each marker has its candidate points established, the algorithm starts calculating which rigid body has the best matching candidates to their assigned markers. Since a marker can have many possible candidates (shown in [Figure 4.4](#)) and a rigid body is composed of various markers, there are many possible combinations of points that fit in a rigid body. For example, looking back to [Figure 4.4](#), if the green and blue marker formed a rigid body, there would be two possible assignments: (2,3) and (2,4). Because of this, the algorithm computes a score function for every possible combination that defines how well an assignment is compared to the structure model of previous frames.

$$C = \sum_{ij} \exp \left(- \frac{(D(i,j) - D'(i,j))^2}{2A(i,j)^2} \right) / N \quad (4.3)$$

[Equation 4.3](#) is the function used to calculate the score C . Matrix D and matrix A are the distance matrix and the standard deviation matrix of the current structure model (discussed in [Subsection 4.2.1](#)); matrix D' is the distance matrix of the candidate structure model if the assignment is made (keeping in mind that all these matrices are updated every frame, so they are the accumulation of all the training frames and the labelled frames until the current frame). N is the number of links present in the rigid body.

Because this step of the algorithm performs an exhaustive search over all the possible combinations, the complexity can escalate quite quickly. It is important that the candidate sets for each marker are not too big, or the algorithm will spend a considerable amount of time calculating the score for every combination.

After calculating the best possible score for every rigid body, the rigid body with the highest score is selected for assignment (becomes satisfied). This means that the combination of points that gave that rigid body the best assignment will be selected, and those points will be permanently assigned to those markers in this iteration. This also means that those points will be removed from other rigid bodies' candidate combinations, and their scores will have to be recomputed, since the optimal assignment might have changed.

This step is performed until no more rigid bodies can be satisfied. The main reason for a rigid body not being able to be satisfied is a missing marker in the candidate set (possibly an occluded marker). When this happens, the algorithm needs to infer the position of the missing markers, which will be explained in the next subsection. This step is also responsible for ruling out any

ghost markers existing in the data because if a ghost marker is selected as a candidate its score will not be better than the score of the actual marker, if it is present in the clip. Because of this, in [Chapter 5](#), the effect of ghost markers will not be explored in great detail.

4.3.3 Filling gaps

The algorithm only performs this step if there are any unset markers during the last phase. If there is a problem in the data, the algorithm, usually during the last iterations of the loop that runs in [Subsection 4.3.2](#), stops its execution because it cannot satisfy any more rigid bodies with the information available. This means that some markers are left without assignment and the algorithm will assume that there are gaps present in the data. In order to fill those gaps, a true-range multilateration algorithm (explained in [Subsection 2.2.5](#)) is utilized to infer the position of the missing points. The markers in a rigid body that contains the missing marker are used as stationary points, and their average distance to the missing marker (taken from the structure model) is utilized as the search radius. With all the markers having one and only one assignment, the algorithm can proceed to perform the actual labelling.

In the case that more than one marker is missing from a rigid body, the algorithm will work as long as there are at least 3 non-missing markers. However, and as expected, the accuracy of the multilateration decreases with the increase of missing markers in a rigid body. In the special case that the rigid body contains less than 3 non-missing markers, the state estimation features of the motion model will be used to infer the position of the missing marker.

So, quickly summing up the gap filling, if a point is missing, its position is inferred either through the rigid body relationships (using multilateration) or the estimation of its associated Kalman Filter.

4.3.4 Re-ordering the frame

The act of "labelling" has, until now, been described as assigning labels to markers. In practice, and due to the nature of a *.c3d* file (explained in [Subsection 4.1.2](#)), what happens is a sorting of the frame's points. All the steps performed until now take the algorithm to the end of step number 2 in [Figure 4.5](#). In this stage, all correspondences between the previous frame and the next frame have been established, so the only thing left to do to "label" the frame is to re-order it according to those correspondences. At this stage, the algorithm can already output a *.c3d* file that is labelled all the way up to the frame that was just processed.

4.3.5 Update Structure and Motion Models

The final step in the algorithm is to prepare the for the next frame. This step consists of adding the new point positions discovered in the labelling to both the structure model and the motion model. This happens, in case the case of the structure model, using the same procedure described in [Subsection 4.2.1](#) but instead of utilizing the nearest neighbour association, the positions provided by the labelling algorithm are used. In the case of the motion model, each Kalman Filter predicts

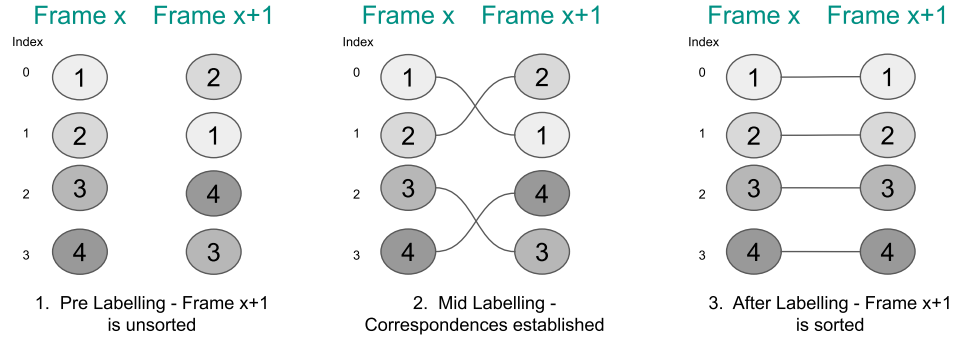


Figure 4.5: Visualization of the index shuffling that occurs when labelling

the position of its associated marker and is then fed with the actual position of the newly labelled points.

Continuously feeding the models with the newly available data keeps them consistent with the motion of the subject and, regarding the Kalman Filters, allows them to change their internal variables to better reflect changes in the type of movement.

4.3.6 Summary of the algorithm

Because the algorithm is rather complex, the pseudocode in [Algorithm 1](#) aims to give a simpler overview of the whole labelling process.

Note that the "Need_Recompute()" attribute is already an optimization applied to the algorithm [Yu et al., 2007]. Whenever a rigid body becomes satisfied, it may change the available candidates of other rigid bodies, rendering their current score invalid. Because of this, rigid bodies that become invalid need to have their maximum score recomputed.

4.4 Testing and Validating

In order to verify the quality of the produced algorithm, the labelled clips need to be evaluated and compared to the ground-truth to analyse the differences between the two. Before labelling the data, though, there is a need to remove the labels from the dataset because it is pointless to label an already labelled clip. There is also a need to define the rules by which the algorithm will be judged. Nuances in the error quantification method can alter the meaning of the results, so a clearly defined evaluation method is mandatory.

4.4.1 Artificially destroying the data

Ideally, testing would be performed in real clips that had natural gaps in them straight from the recording software, however, these types of recordings are not readily available online, so there is a need to convert the ground-truth, labelled clips into more realistic *.c3d* files that would come

Algorithm 1 The full labelling algorithm in pseudocode**Input:** Unlabelled frame, Motion and Structure Model (mm, st), Rigid bodies (rb)**Output:** Labelled frame

```

1:  $Status(rb_i) = "unsatisfied"$  and  $Need\_Recompute(rb_i) = TRUE$  for  $i = 1...LEN(rb)$ 
2: Based on the motion model, get a set of candidates for each marker – Subsection 4.3.1
3: while at least one rigid body is "unsatisfied" do
4:   for each rigid body  $rb_i$  do
5:     if  $Status(rb_i) = "unsatisfied"$  AND  $Need\_Recompute(rb_i) = TRUE$  then
6:       Exhaustively search for the best assignment – Subsection 4.3.2
7:        $Need\_Recompute(rb_i) = FALSE$ 
8:     end if
9:   end for
10:   $best\_rb = \max(Score(rb))$   $\triangleright$  Rigid body with the best assignment – Equation 4.3
11:   $Status(best\_rb) = "Satisfied"$ 
12:  for each "unsatisfied" rigid body  $rb_i$  do
13:    if  $Assignments(best\_rb)$  intersects  $Assignments(rb_i)$  then
14:       $Candidates(rb_i) = Candidates(rb_i) - Assignments(best\_rb)$ 
15:       $Need\_Recompute(rb_i) = TRUE$ 
16:    end if
17:  end for
18: end while
19: Fill gaps trough Multilateration or Kalman Filter Estimation – Subsection 4.3.3
20: Re-order frame – Subsection 4.3.4
21: Update Structure and Motion Model – Subsection 4.2.1 and Subsection 4.2.2

```

straight from the MoCap software. For this purpose, there are two steps that need to be taken: removing the labels from the data and creating gaps.

As was explained in [Subsection 4.1.2](#), a *.c3d* file contains an ordered list of points for every frame, and the order of the points dictates its labelling. In order to "unlabel" a clip, the points inside each frame are shuffled randomly. By removing the order relationship between the points of every frame and the order of the list of labels, the clip is deprived of any relationship between its labels and its markers.

To simulate gaps that naturally occur in real MoCap acquisitions (due to the occlusion of markers) some markers need to be removed from the clip. Artificially removing markers in a way that resembles what happens in real videos is a topic that could be studied on its own, due to the somewhat unpredictable nature of occlusions. Two approaches were designed in order to test the quality of the gap filling algorithms. The first approach removes a random marker, every N frame, during a certain amount x of frames. The values of N and x can be tuned in order to evaluate the effect of the gaps in the gap filling method. For example, in one of the cases studied every 240 frames, the clip had a 10 frame gap in some random marker – the testing was performed with 48 different combinations of N and x , as shown in [Section 5.3](#). The second, slightly more thoughtful approach (despite being quite qualitative), involves visually inspecting the clip and manually removing markers in strategic positions to test for occlusions during two scenarios:

- **Brisk Gaps** – The marker was in an unpredictable state during the gap. If a marker is following a constant movement and is removed right before a sharp turn in its trajectory, the system's only way of detecting that change is through the other markers in the same rigid body, which may not reflect perfectly the missing marker position and sometimes the recovery of markers through the rigid body relationships is impossible. This leaves the inference to be made only by the Kalman Filter, whose knowledge is that the marker was performing a constant movement (and this knowledge will not change unless the measurements tell otherwise).
- **Smooth Gaps** – The marker was and stayed in a predictable trajectory during the gap. If a marker maintains a smooth movement during a long span of frames (for example with constant velocity), removing it for some frames during that span of frames, theoretically, allow the Kalman Filter alone to generate a good estimation of the position. These types of gaps, considered in this dissertation to be "smooth", are expected to be filled.

These two gap generation method, although surely not perfect, can give some indication about the impact of different types of gaps in the overall quality of the labelling.

To match the methodology of [Yu et al., 2007] and thus have a fairer ground of comparison – while still not perfect since there is no access to Yu's algorithm or Yu's MoCap clips – labelling was performed in clips that were resampled. Resampling in this context mean reducing the frames per second of a recording down from its original rate. Reducing, for example, a 120fps clip to only 60fps implies removing half of its marker information and, in a way, creating a 1 frame gap every other frame. This makes the labelling more challenging and is, in a way, similar to the periodic gap introduction.

4.4.2 Error Quantification

The labelling of a clip with the developed method can happen in one of two ways: either the matching marker in frame $x + 1$ is found and its position is used, or the marker is missing from the data and its position is inferred, either through multilateration or using the Kalman Filter estimated position.

Because of the second option, there can be slight differences between the inferred position and the real position, and even the smallest of deviations will not count as a correct match if the comparison is done by exact comparison, point by point. To prevent this issue, the comparison is allowed to consider correct every point that lies within a range e of the ground truth, with e being the acceptable amount of error. With a defined method to classify if a point is in the correct position or not, a clip's overall quality can be measured by the ratio of correctly labelled points over the total amount of labelled points as defined in Equation 4.4, with a *Quality* of 1.0 meaning that all points lie within range e of their ideal position.

$$Quality = \frac{CorrectLabels}{TotalLabels} \quad (4.4)$$

Additionally, subtracting the labelled clip from the ground-truth yields a distance measure that represents the error of the labelling. The average and standard deviation of this error can be taken individually for each marker or globally for the whole clip, and is a good indicator of which markers are failing more, when they are failing and the type of movement they were performing when they failed.

Chapter 5

Results and Discussion

When evaluating a complex system such as the one developed, it is not enough to evaluate it only by its overall performance. There are several factors that may or may not contribute to the performance of the system, and the effect of the different implemented alternatives should be studied in order to figure out what works and what does not. This chapter will be dedicated to analysing the results in different scenarios, exploring the impact of each scenario on the overall performance.

5.1 Labelling

A good starting point for the analysis is the comparison of the labelling using Kalman Filters versus using Unscented Kalman Filters. These comparisons were made in gap-less clips, in order to isolate the effect of the type of filter. All measurements were taken with an acceptable error range of 5 millimetres. Also present in the graph are labelling performances done by Nearest Neighbour associations that serves as a base for comparison. As can be seen in [Figure 5.1](#), for the Kalman filter, its lowest performance yielded a labelling with 57.31% of correct labels and in its highest performance it achieved a perfect 100% correct labels, having an average of 87.07% correct labellings. As for the Unscented Kalman Filter, its performance averaged 90.41% of correct labels, with a minimum of 66.37% and a maximum of 100%. There are however some outliers in the results, like the file "143_42" in which the UKF achieved 100% of correct labels versus the 83.59% of the regular Kalman Filter.

The left graph of [Figure 5.2](#) compares the length of a clip, in frames, to the time it took to label it. The relationship is somewhat linear, with every clip taking on average 4.41 times its length in seconds to label. No relationship was found between the labelling time and the performance, but there is a relationship between the time to label and the number of points that need to be inferred (mostly due to the expensive multilateration algorithm). The graph on the right in [Figure 5.2](#) shows the labelling performance in frames per second, and matches the graph on the left with a mostly constant number of frames labelled per second, averaging 15.2 frames labelled per second.

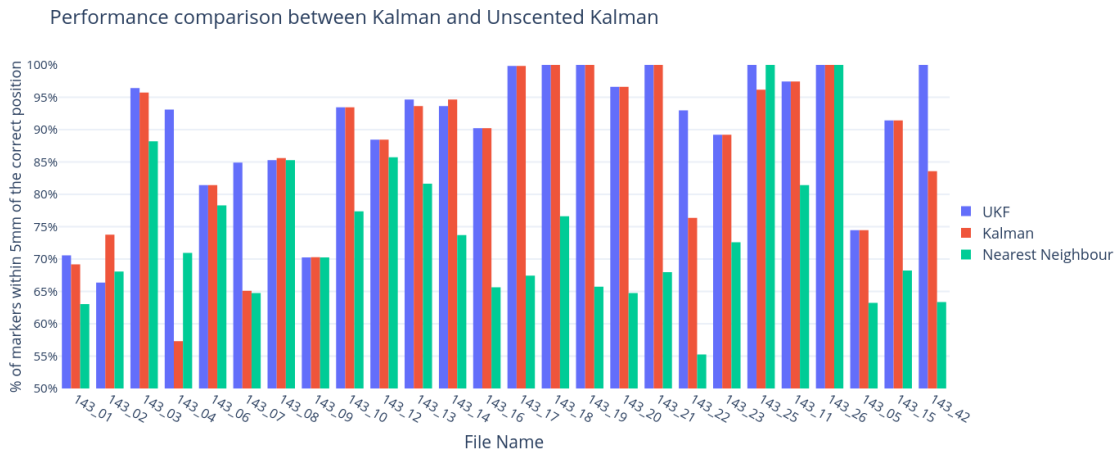


Figure 5.1: Comparison of the performance between Kalman Filters and Unscented Kalman Filters and Nearest Neighbour.

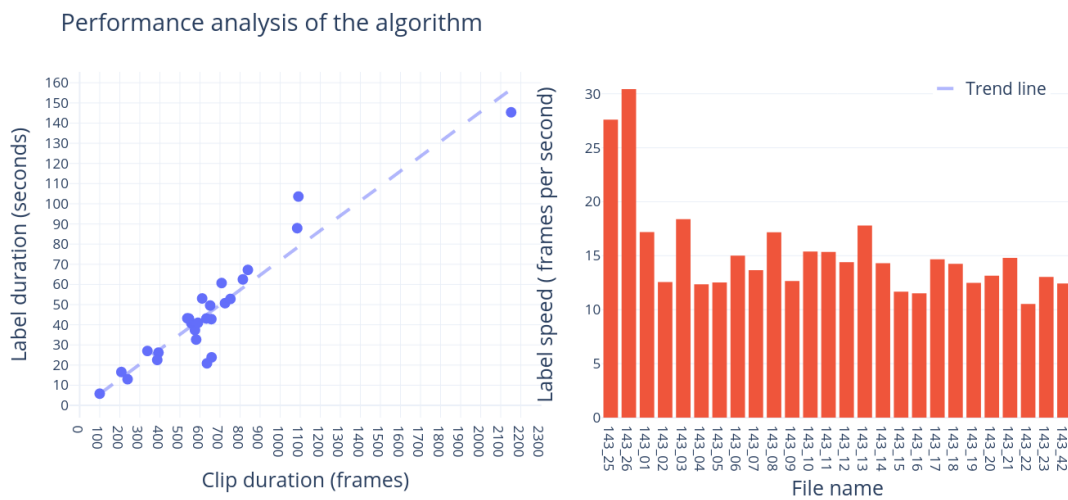


Figure 5.2: Influence of the clip length (in frames) on the labelling time taken across multiple files, with the corresponding trend line.

5.1.1 Profiling

In order to figure out where the system spends the most time, a profiling analysis was performed and, in [Figure 5.3](#) the results are very clear. Naturally, most of the system's time is spent labelling the clip – 99.9% of the program occurs in the labelling.py file. However, during the labelling, 94.51% of that time is spent updating the Motion Model. This profiling was performed in a gap-less version of the "143_25" file using Unscented Kalman Filters.

This happens because updating the motion model requires updating all the Kalman Filters, which includes expensive computations for every marker. Blocks A and B in [Figure 5.3](#) refer to the second and third most expensive operations in the algorithm, which are the exhaustive search step of calculating the best assignment and the selection of candidates that involves the creation of a KD-Tree and searching it for the best candidate of each marker.

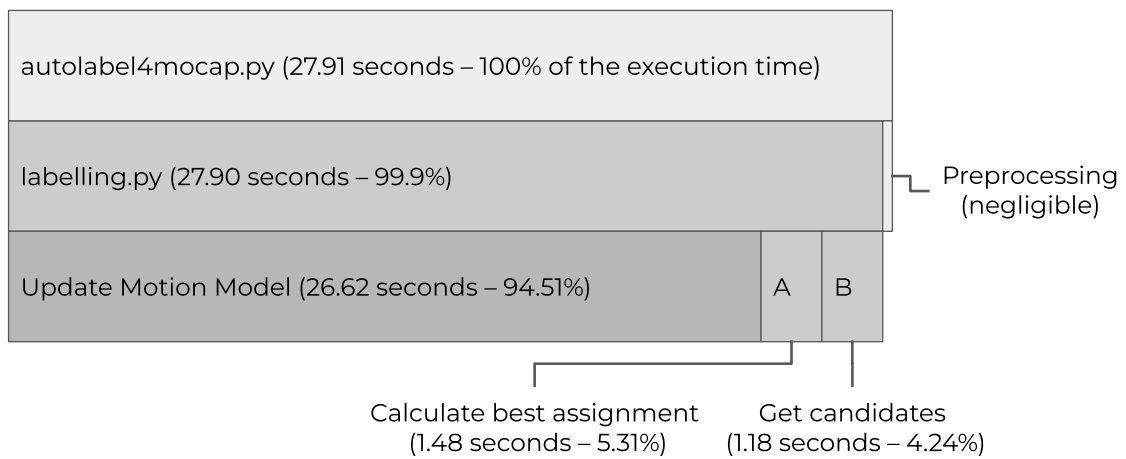


Figure 5.3: Visualization of the profiling of the full running system. Blocks are not to scale.

5.2 Rigid Body Fitting

The grouping of markers into rigid bodies is expected to have an impact in the labelling performance, so it makes sense to study the developed grouping method and compare it to the performance of manually input rigid bodies. In [Figure 5.4](#), a comparison was made between the performance of the algorithm utilizing automatically fitted rigid bodies (with 4 markers per rigid body, though this value can be changed by the user) and manually input rigid bodies. All the label attempts use the same configurations — Gap-less files using the UKF approach — to isolate the effect of the rigid body fitting. The Automatic Fitting actually performed very similarly to the Manual Input, each having an average percentage of correct marker of 90.90% and 90.41%, respectively.

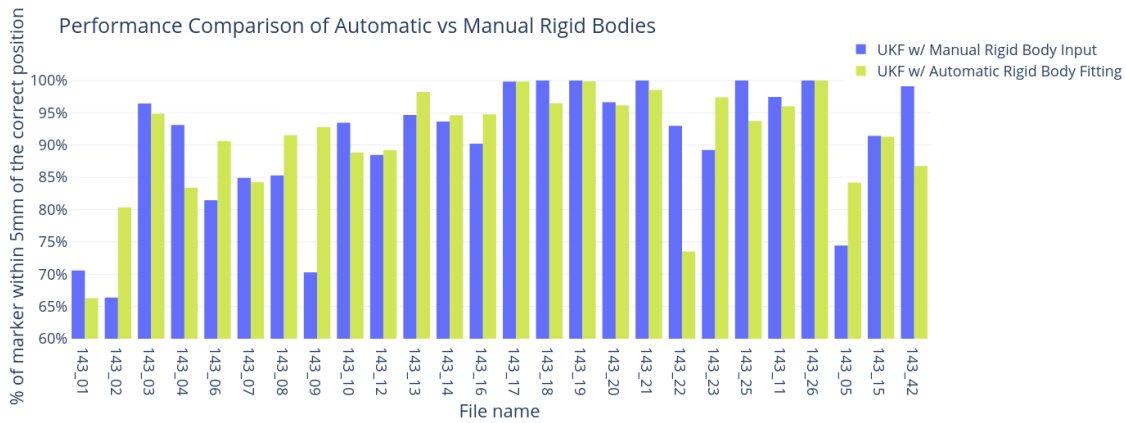


Figure 5.4: Performance Comparison of Automatic vs Manual Rigid Bodies

5.3 Gap-filling

Analysing, first, the automatic introduction of gaps periodically in the data, [Figure 5.5](#) shows the relationship between the percentage of correct labels in the full clip and the gap size with each file having gaps introduced at different intervals (represented by the same colour markers at each gap size). A quick analysis of the graph shows that there isn't much of a relationship between the gaps introduced automatically in the clips and the quality of the labelling. Because of this hard to analyse data, this approach was not further explored. On the other hand, looking at [Figure 5.6](#),

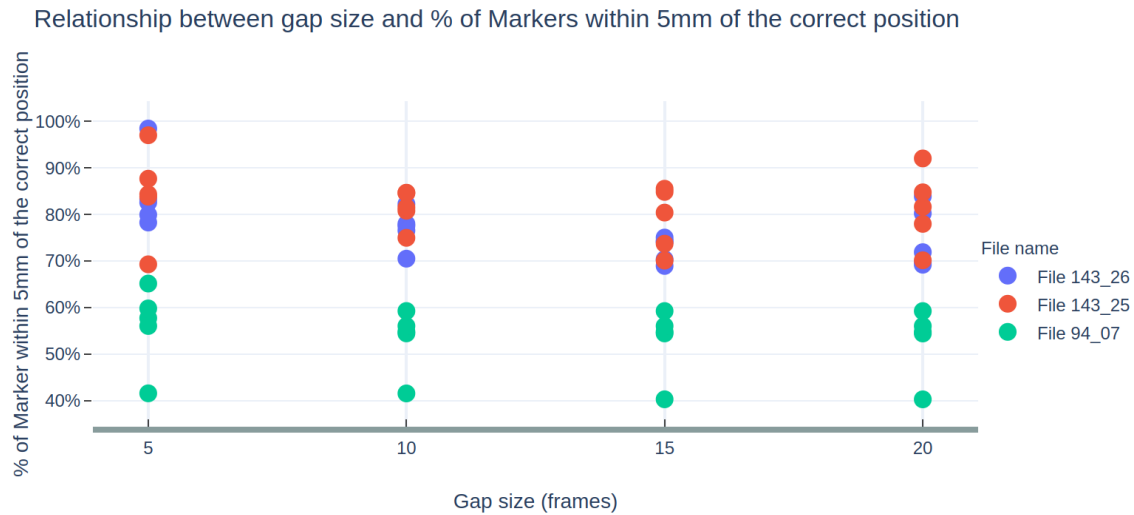


Figure 5.5: Comparison of the performance between different gap sizes introduced periodically in the data.

when gaps are placed in strategic places, there is a clear decline in quality with the increase of the gap size as well as a clear distinction between gaps introduced during brisk movements and gaps introduced in smooth movements. The files "143_25" and "143_26" were chosen for testing

due to their 100% perfect labels in the gap-less tests, removing the possible labelling flaws in this analysis. The file "94_07" was chosen because the clip contains a lot of fast, erratic movements and is considered to be a difficult to label file. The purpose of the connected lines in the graph is to show the trend of each individual file and gap type – data was only collected for gap sizes multiple of five. Despite being a rather simple analysis, it shows that the algorithm is affected by missing markers, with, for example, the files "143_25" and "143_26" dropping from a perfect labelling with 100% accuracy with no gaps to as low as approximately 70% correct labels when 20 frame gaps are introduced in brisk movement areas.



Figure 5.6: Comparison of the performance between different gap types and sizes introduced manually in the data. Dashed and full lines correspond to Brisk and Smooth Gaps, respectively.

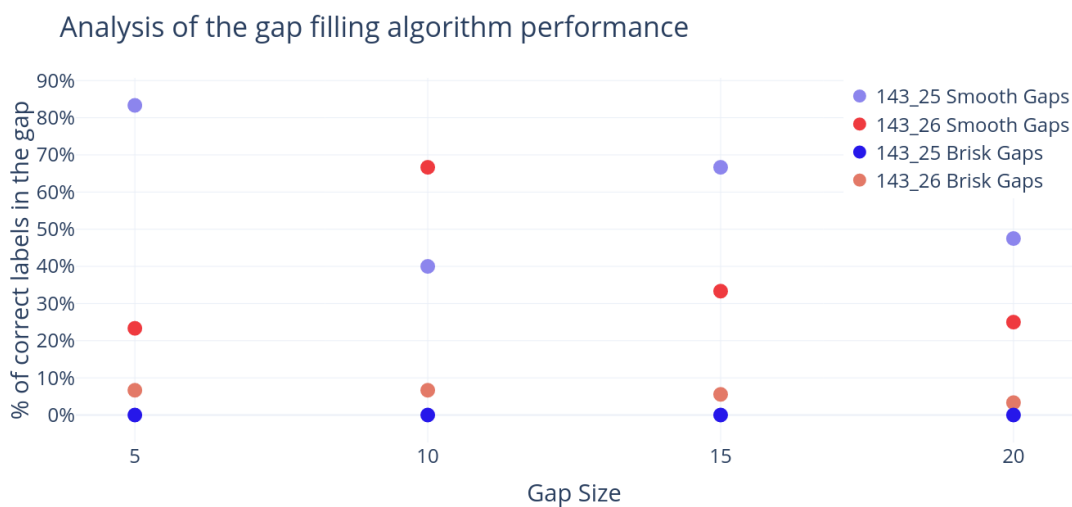


Figure 5.7: Analysis of the gap filling algorithm performance

In Figure 5.7, the comparison was only made in the gap itself. While the previous analysis compared the full labelled clip with the ground truth, this one only explores how well a gap was filled – compares to the ground truth but only in the gap areas. The results shown for each file are an average of all the gaps introduced in the file. The results match the previous full clip analysis, with smooth gaps being filled with a higher percentage of correct labels than brisk gaps. This analysis, however, demonstrates that some gaps are simply not filled correctly – the file "143_25" with brisk gaps has exactly zero points filled correctly in all of its gaps.

5.3.1 Resampling

The Figure 5.8 shows the labelling results of the down sampled clips. As was expected, reducing the FPS, has a negative impact in the labelling quality. Some very harsh cases are for example the files "143_01" and "143_02" that dropped from 71% and 61% to lower than 16% in both down samplings.

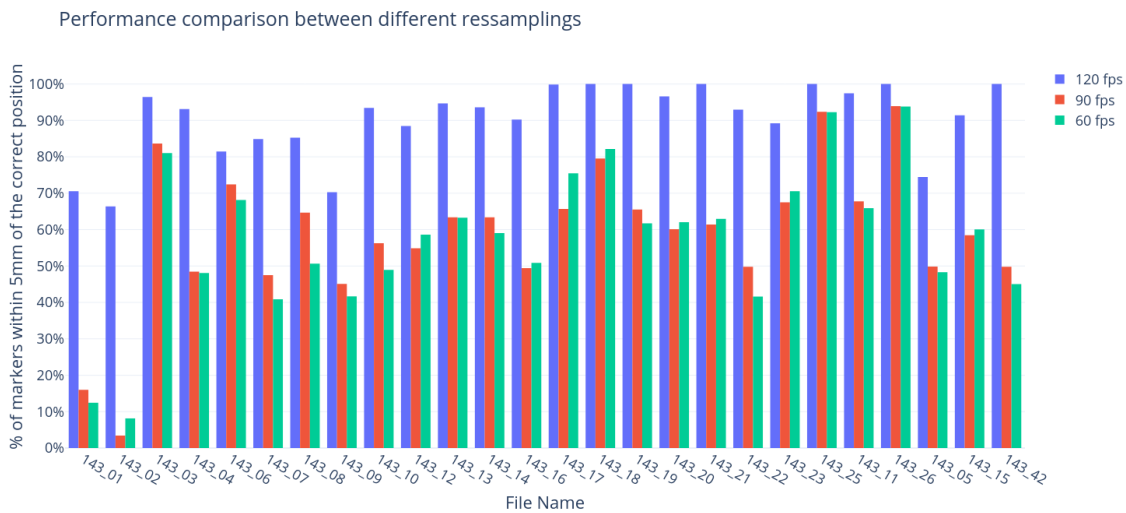


Figure 5.8: Performance comparison between different resamplings

5.4 Discussion

In summary, this method is a tracking algorithm with exhaustive search characteristics that assimilate themselves to a brute force approach. The algorithm utilizes various techniques (aided by the Motion and Structure model) to reduce the scope of the exhaustive search, in order to transform the brute-force matching of markers into a more focused search between markers that could be mistaken for each other.

The biggest challenge that the algorithm has to accomplish is to correctly track the markers and estimate as correctly as possible the marker future positions. If the motion model is correctly set up and follows the markers as closely as possible, the candidates for each marker in each frame

are fewer and more accurate, leading to a faster exhaustive search (fewer possible combinations) which in turn results in a fast labelling algorithm. Properly set up Kalman Filters also reduce the chance that a marker goes missing due to erratic movements – bad Kalman Filters that are too far away from the actual position need a very large search radius to find the true candidate and usually end up with no valid candidate.

5.4.1 Fitting

As can be seen by the results, the developed rigid body fitting method closely matches the manually input rigid bodies. This is a good indicator of the quality of the method, proving that, in the tested scenarios, this somewhat naïve method did not interfere in the quality of the labellings, however there are some things that need to be kept in mind when using this method. This first disadvantage is the dependency on the movement in the training frames. In the extremely edge case that the subject stays completely stationary during the training frames, the standard deviation of the distance of all markers to each other will be the same – zero. This means that the algorithm can choose any n markers to group together since they are all, as far as the system knows, at a constant distance between each other. This can cause the algorithm to group together markers that have no real relationship and thus ruin the labelling algorithm.

Another possible problem is that the method forces all the rigid bodies to have the same number of markers. If the number of markers per rigid body is set to a higher number than what is necessary and plausible for the subject captured, the rigid bodies will contain unnecessary markers that will destabilize the rigidity of the relationships inside the rigid bodies.

5.4.2 Gap Filling

The data shown in [Section 5.3](#) was clear to show that the type of movement the marker is performing is a crucial factor in how well its position is estimated. Introducing gaps in random markers at arbitrary frames yielded labelling performances just as random because the trajectory of the marker before and after the gap highly affects its labelling. The internal state of the Kalman Filter needs some time to adjust to a marker's trajectory, and once the marker goes missing, it has no information about a trajectory change. This leads the system to extend the current trajectory for the gap duration, which is not what actually happens to the marker.

The rigid bodies that contain the marker are also used to infer the position of the missing marker whenever possible, and usually perform a better estimation than the Kalman Filter alone. However, in the current system, the use of the rigid body estimation is limited to rigid bodies that have at least 3 non-missing markers (to ensure a proper multilateration).

It is to note that the tests performed are considered to be of high difficulty. Introducing 20 frame gaps obscures a considerable amount of movement information that the system has to "guess". The same happens with Yu's method of evaluation that down samples the data, some-time to half the original rate.

5.4.3 Performance

Due to the ease of its ease of use and fast prototyping capabilities, the system was developed in Python. This means that, unfortunately, the system's overall performance is hindered by python's naturally slower processing time. Even with the use of libraries like Numpy that use pre-compiled C code, the performance still is not comparable to that of systems written purely in higher performance (but rather low-level) languages like C and C++. The profiling of the system showed that the Kalman Filters take up a significant percentage of the program's execution time. One simple cause for this is that the FilterPy library (used for its Kalman Filters implementation) was not made with performance in mind, but rather as a means to teach about Kalman and other types of filters. On a quick note about ghost markers, they are not expected to have influence in the overall labelling performance of the algorithm since they will most likely be ignored, but if they are present in great number in a small area they can slow down the algorithm due to the high number of possible candidates of a marker.

The results of the performance analysis show that the developed system is not on par with current state-of-the-art technology that can operate in real-time. However, the results are still in quite an acceptable range and the system is considered to be operating in a reasonable amount of time. Due to its complexity, the proposed objective of guiding the user through the manual correction process was not implemented. However, although not thoroughly explored, the system has the capabilities of pointing out the areas of most uncertainty – which was also a valued property of the algorithm. This information could be compiled into a tool that guides the user through the manual labelling and is considered a valuable future work research.

Chapter 6

Conclusion and Future Work

As was shown in [Chapter 3](#), the existing methods for the automatic labelling of MoCap clips are many and very diverse in their approaches. This dissertation, while basing itself on [\[Yu et al., 2007\]](#), introduced a new implementation that studied different aspects of the algorithm more in-depth and applied to a different dataset. Even though a direct comparison with Yu’s method is not feasible, the implemented method’s performance is not on par with what was described in [\[Yu et al., 2007\]](#). This dissertation started with the goal of producing a robust algorithm that maps labels to markers in the presence of occlusions and ghost markers, and the developed system can successfully accomplish these tasks in various tested scenarios. It cannot be said though that it is robust and perfectly does what it set out to do, having some problems especially in files with fast, erratic movements. However, it is still a starting point and with a lot of room for improvement in various parts of the algorithm. The developed approach does not require any special initialization pose or expensive training of Neural Networks (unlike other state of the art methods) and can be fully automatic and input-less (assuming the default values are sufficient), achieving perfect labellings (and averaging 90.41% of correct labels in its best configuration) without any previous knowledge of the motion and skeleton type.

6.1 Future Work

There are several solutions that could be implemented in the system and their results, although uncertain, could yield better results and/or faster performance. For example, regarding the fitting of rigid bodies, the automatic fitting algorithm could be used as only a baseline and manually adjusted if need be; other grouping algorithms could be experimented with like a clustering algorithm such as K-Means or a similarity graph like the method presented in [\[Mu Lee et al., 2013\]](#); additionally, the rigid bodies could be re-fitted in the middle of labelling the clip to account for any mistake in the initial fitting, either periodically or using some metric gathered by the system, namely the number of unsatisfied rigid bodies at the end of a frame or a certain threshold in the standard deviation of markers in a rigid body. Furthermore, in the case of marker inference, instead of using only the estimation of the Kalman Filters or the relationships of the rigid body in

a multilateration context, new techniques could be explored to make use of all the data available, for instance the relative position of the markers inside a rigid body (the direction vector) is never used. The algorithm can also be made a lot faster by programming techniques alone – rewriting the algorithm in another, more performing language or parallelizing the code in areas such as the exhaustive search – and, taking into account the information provided by the profiling done, a lot more care could go into using faster Kalman Filter implementations.

References

- [Alexanderson et al., 2017] Alexanderson, S., O’Sullivan, C., and Beskow, J. (2017). Real-time labeling of non-rigid motion capture marker sets. *Computers & Graphics*, 69:59–67.
- [Bregler, 2007] Bregler, C. (2007). Motion Capture Technology for Entertainment [In the Spotlight]. *IEEE Signal Processing Magazine*, 24(6):160–158.
- [Clouthier et al., 2021] Clouthier, A. L., Ross, G. B., Mavor, M. P., Coll, I., Boyle, A., and Graham, R. B. (2021). Development and Validation of a Deep Learning Algorithm and Open-Source Platform for the Automatic Labelling of Motion Capture Markers. *IEEE Access*, 9:36444–36454.
- [Edwards, 2016] Edwards, K. (2016). Researchers incorporate motion analysis technology into health care | UMKC School of Medicine.
- [Furniss, 1999] Furniss, M. (1999). MIT - Motion Capture. <http://web.mit.edu/comm-forum/legacy/papers/furniss.html> (last access: January 2022).
- [Ghorbani et al., 2019] Ghorbani, S., Etemad, A., and Troje, N. F. (2019). Auto-labelling of Markers in Optical Motion Capture by Permutation Learning. *arXiv:1907.13580 [cs]*, 11542:167–178. arXiv: 1907.13580.
- [Gomes et al., 2021] Gomes, D., Guimarães, V., and Silva, J. (2021). A Fully-Automatic Gap Filling Approach for Motion Capture Trajectories. *Applied Sciences*, 11(21):9847. Number: 21 Publisher: Multidisciplinary Digital Publishing Institute.
- [Guerra-Filho, 2005] Guerra-Filho, G. (2005). Optical Motion Capture: Theory and Implementation. *RITA*, 12(2):61–90. Publisher: Citeseer.
- [Jiménez Bascones et al., 2019] Jiménez Bascones, J. L., Graña, M., and Lopez-Guede, J. M. (2019). Robust labeling of human motion markers in the presence of occlusions. *Neurocomputing*, 353:96–105.
- [Johnson-Glenberg and Megowan-Romanowicz, 2017] Johnson-Glenberg, M. C. and Megowan-Romanowicz, C. (2017). Embodied science and mixed reality: How gesture and motion capture affect physics education. *Cognitive Research: Principles and Implications*, 2(1):24.
- [Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, pages 182–193. SPIE.
- [Kaiga et al., 2007] Kaiga, T., Yukawa, T., Mitobe, K., Miura, T., Tamamoto, H., and Yoshimura, N. (2007). Magnetic motion capture system measuring movements of hands and a body simultaneously. In *ACM SIGGRAPH 2007 posters*, SIGGRAPH ’07, pages 79–es, New York, NY, USA. Association for Computing Machinery.

- [Kalman, 1960] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45.
- [Kim and Kim, 2014] Kim, J. and Kim, M. (2014). Motion capture with high-speed RGB-D cameras. In *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 394–395. ISSN: 2162-1241.
- [Kim and Bang, 2018] Kim, Y. and Bang, H. (2018). *Introduction to Kalman Filter and Its Applications*. IntechOpen. Publication Title: Introduction and Implementations of the Kalman Filter.
- [Krogh, 2008] Krogh, A. (2008). What are artificial neural networks? *Nature Biotechnology*, 26(2):195–197. Number: 2 Publisher: Nature Publishing Group.
- [Kuhn, 1955] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1):83–97.
- [Malus et al., 2021] Malus, J., Skypala, J., Silvernail, J. F., Uchytíl, J., Hamill, J., Barot, T., and Jandacka, D. (2021). Marker Placement Reliability and Objectivity for Biomechanical Cohort Study: Healthy Aging in Industrial Environment (HAIE—Program 4). *Sensors*, 21(5):1830.
- [Merwe et al., 2004] Merwe, R., Wan, E., and Julier, S. (2004). Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion—Applications to Integrated Navigation. *Proceedings of the AIAA Guidance, Navigation & Control Conference*, 3.
- [Meyer et al., 2014] Meyer, J., Kuderer, M., Müller, J., and Burgard, W. (2014). Online marker labeling for fully automatic skeleton tracking in optical motion capture. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5652–5657. ISSN: 1050-4729.
- [Mu Lee et al., 2013] Mu Lee, K., Matsushita, Y., Rehg, J. M., and Hu, Z. (2013). *Computer vision*. 2. Number 7725 in Lecture notes in computer science. Springer, Berlin Heidelberg.
- [Nakano et al., 2020] Nakano, N., Sakura, T., Ueda, K., Omura, L., Kimura, A., Iino, Y., Fukashiro, S., and Yoshioka, S. (2020). Evaluation of 3D Markerless Motion Capture Accuracy Using OpenPose With Multiple Video Cameras. *Frontiers in Sports and Active Living*, 2.
- [Pachipala et al., 2018] Pachipala, Y., Bharadwaj, C., Sai, K., and Narasimham, C. (2018). Implementing Robots in Defence Through Motion Capture with Mixed Reality. *International Journal of Engineering and Technology(UAE)*, 7:114–116.
- [Pueo and Jimenez-Olmedo, 2017] Pueo, B. and Jimenez-Olmedo, J. (2017). Application of motion capture technology for sport performance analysis. *Retos: nuevas tendencias en educación física, deporte y recreación*, 2017:241–247.
- [Radoff, 2008] Radoff, J. (2008). Anatomy of an MMORPG. <https://web.archive.org/web/20091213053756/http://radoff.com/blog/2008/08/22/anatomy-of-an-mmorpg/> (last access: January 2022)}.
- [Rahul M., 2018] Rahul M. (2018). Review on Motion Capture Technology. *Global Journal of Computer Science and Technology*, 18(1):5.

- [Reverdy et al., 2015] Reverdy, C., Gibet, S., and Larboulette, C. (2015). Optimal marker set for motion capture of dynamical facial expressions. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*, pages 31–36, Paris France. ACM.
- [Reynolds, 2009] Reynolds, D. A. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663). Publisher: Springer City.
- [Xia et al., 2017] Xia, S., Su, L., Fei, X., and Wang, H. (2017). Toward accurate real-time marker labeling for live optical motion capture. *The Visual Computer*, 33(6):993–1003.
- [Yu et al., 2007] Yu, Q., Li, Q., and Deng, Z. (2007). Online Motion Capture Marker Labeling for Multiple Interacting Articulated Targets. *Computer Graphics Forum*, 26(3):477–483. _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2007.01070.x>.