Estimation of Human Poses with mmWave Radar Sensors

Master's Thesis in Artificial Intelligence

submitted
by

Eduardo Javier Feria Rendón

born 01.11.1995 in Las Tunas, Cuba

Written at

Machine Learning and Data Analytics Lab
Department Artificial Intelligence in Biomedical Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

Advisors: M. Sc. Lukas Engel, Dr. Eva Dorschky, M. Sc. Daniel Krauß, Prof. Dr. Björn Eskofier & Prof. Dr. Martin Vossiek

Started: 01.09.2023

Finished: 29.04.2024

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.
Die Richtlinien des Lehrstuhls für Bachelor- und Masterarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Erlangen, den 29. April 2024

Acknowledgment

First and foremost, I am very grateful to God for all the blessings and opportunities He has given me.

Thanks to my wife, Susi, who is largely responsible for my success and has stood by me throughout the whole process of this thesis.

I am very grateful to my advisors M. Sc. Lukas Engel, Dr. Eva Dorschky, M. Sc. Daniel Krauß, Prof. Dr. Björn Eskofier and Prof. Dr. Martin Vossiek. Thank you for the opportunity to be part of this thesis and for all the support you have given since the beginning.

Thanks to Markus Bergmann and to all participants of this study.

Thanks to Nehemiah Gateway, especially Arlinda Merdani, Margit Lange and Arnold Geiger, without your support I probably would not have been able to fulfil my dream of studying AI.

Thanks to Markus Decker for helping me with proofreading corrections.

The gratefully acknowledge the scientific support and High Performance Computing (HPC) resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). The hardware is funded by the German Research Foundation (DFG).

Übersicht

Das Erfassen des Wesens der menschlichen Bewegung eröffnet Möglichkeiten von der Mensch-Computer-Interaktion bis zur medizinischen Diagnose. Human Pose Estimation (HPE) als Aufgabe transformiert Daten wie RGB- oder Tiefenbilder in eine Posenbeschreibung um, die in der Regel eine Reihe von Schlüsselpunkten umfasst, die ein menschliches Skelett bilden. HPE wird in der Regel mit RGB-Kameras durchgeführt, die ein erhebliches Datenschutzproblem darstellen und gleichzeitig zu Ungenauigkeiten in der Tiefe führen und anfällig für Änderungen der Lichtverhältnisse sind. Andererseits liefern Radare im Millimeterwellenbereich genaue Informationen über über Standorte und Zielgeschwindigkeiten, was sie zu einer natürlichen Wahl für HPE-Aufgaben macht.

Umfassende HPE-Datensätze, die Radar-Punktwolken enthalten, sind selten, und die meisten von ihnen enthalten nur wenige Teilnehmer oder weisen Ungenauigkeiten in den Berechnungen der Skelettpunkte auf. Um einen Beitrag zur zukünftigen Forschung auf diesem Gebiet zu leisten, stellt diese Arbeit den mmRadPose-Datensatz für HPE und ein Datenerfassungsprotokoll zu dessen Erweiterung vor. Der Datensatz umfasst mehr als 200k Tupel von 7-dimensionalen radargenerierten Punktwolken und 26-Schlüsselpunkt-Skelette. Zwölf Teilnehmer nahmen an der Studie teil und führten elf Bewegungen aus in drei verschiedenen Winkeln in Bezug auf das Radar (dem Radar zugewandt, 45° gegen den Uhrzeigersinn gedreht und 90° gegen den Uhrzeigersinn gedreht). Der Datensatz enthält auch sequenzielle Daten.

Ergänzend werden zwei Deep Learning (DL)-Architekturen vorgeschlagen, die auf Pointnet [Qi17a] basieren: PointnetPose, ein Modell zur Verarbeitung von Einzelbildern HPE, und PointnetPoseRNN, das in der Lage ist, mmWave-Radar-Punktwolken-Sequenzen zu verarbeiten. Beide Modelle können Punktwolken ohne vorherige Transformation verarbeiten und sind auf die Verarbeitung von Punktwolken mit Punkten beliebiger Anzahl von Dimensionen zugeschnitten. Beide Modelle wurden an Daten aus dem mmRadPose-Datensatz mit Aufnahmen von der Teilnehmer mit Blick auf das Radar. Das PointnetPose Modell zeigte einen Mean Per Joint Position Error (MPJPE) von 13.62 cm, während PointnetPoseRNN einen MPJPE von 11.96 cm auf Sequenzen der Länge 10 erreicht. Schließlich wurde ein PointnetPoseRNN-Modell für Sequenzen der Länge 10 auf dem gesamten Datensatz trainiert, trainiert, wobei die Teilnehmer in alle drei Richtungen blickten und ein MPJPE von 9.94 cm erreichten.

Abstract

Capturing the essence of human motion opens up possibilities from human-computer interaction to medical diagnosis. HPE as a task, transforms data such as RGB or depth images into a pose description, typically comprising a set of keypoints that form a human skeleton. HPE is typically conducted via RGB cameras, which pose a significant privacy issue while also introducing depth inaccuracies and vulnerability to changes in lighting conditions. On the other hand, radars in millimeter-wave range provide accurate information about locations and target velocities, making them a natural choice for HPE tasks.

Comprehensive HPE datasets containing radar pointclouds are rare, with most of them containing few participants or presenting inaccuracies in the skeleton keypoint calculations. To contribute to future research in the field, this thesis introduces the mmRadPose dataset for HPE and a data recording protocol to extend it. The dataset comprises more than 200k tuples of 7-dimensional radar generated pointclouds and 26-keypoint skeletons. Twelve participants took part in the study performing eleven movements facing three different angles with respect to the radar (facing the radar, rotated $45°$ counterclockwise, and rotated $90°$ counterclockwise). The dataset also features sequential data.

Complementary, two DL architectures based on Pointnet [Qi17a] are proposed: PointnetPose, a single frame processing HPE model, and PointnetPoseRNN, capable of process mmWave radar pointcloud sequences. Both models can process pointclouds without any previous transformation and are tailored to handle point clouds with points of any number of dimensions. Both models were tested on data from the mmRadPose dataset with recordings of the participants facing the radar. The PointnetPose model showed an MPJPE of $13.62\,\text{cm}$, while PointnetPoseRNN achieves an MPJPE of $11.96\,\text{cm}$ on sequences of length 10. Finally, a PointnetPoseRNN model on sequences of length 10 was trained on the entire dataset, with participants facing in all three directions, achieving an MPJPE of $9.94\,\text{cm}$.

Contents

Chapter 1

Introduction

1.1 Motivation

Machine Learning (ML) applications are highly integrated in everyday life. Recommendations systems, automatic text translations, or performing an internet search from an image are some applications all of us have used. One of the topics that has caught the attention of researchers is the analysis of human movement. Automated human movement analysis has a profound impact in sports [Lyu24; Che23; Fu23], health care [Zhu24b; Xu22; Mar22], surveillance [Che18], home automation, automotive industry [Gei16], audiovisual content generation, etc. HPE [Ben21] serves as a framework for analyzing human movement and helps with a number of tasks such as human computer interaction, gesture recognition and activity recognition [Liu23].

HPE tasks attempt to infer the position of the human body in 2D [Uhl22; Cao19] or 3D [Zho24] space. The human pose can be represented simply as a skeleton by detecting specific body keypoints like head, toes, elbows, etc.

Currently, most of HPE applications derive their data from RGB cameras [Cao19; Luo22; Zho24; Den24]. However, the audiovisual sensor technology poses a strong intrusion into people's privacy. In this respect, Millimeter wave (mmWave) radar sensors offer a decisive advantage since the data does not have any immediately recognizable personal reference [Has24]. Unlike cameras, radar sensors maintain robustness against variations in light conditions. Consequently, for both private and public buildings, radar sensors present numerous appealing opportunities [Zhu24a] for performing tasks which involve human behavior, even with high data protection requirements. mmWave radars have proven to be useful for people detection, pose estimation [Sen20; Lee22], gesture recognition [Yan23a] and even to measure vital parameters [Mer19].

mmWave radars operate on a frequency between 30 and 300 GHz, which results in having a wavelength in the millimeter range. This sensitivity allows them to detect movements on the scale of fractions of a millimeter. Their precision, power efficiency, along with their affordable price and compact size have made them a viable option for diverse applications [Koc22; Mow22].

The goal of this thesis is to conduct HPE utilizing mmWave radar point clouds. These point clouds include not only the spatial 3D coordinates but also supplementary attributes provided by the radar. The HPE problem is classically formulated as a regression problem [Tos14]. The input data can be images, videos, pointclouds or radar data, while the ground truth is some form of skeleton representation formed by body keypoints. The estimation of human poses is enabled by determining the positions of keypoints of the human body such as the head, shoulders, arms, legs and joints. HPE constitute is a challenging task, it is sensitive to data with occluded body parts, making it specially hard for the algorithm to estimate the location of the keypoints.

In this work, a 60 GHz frequency modulated continuous wave (FMCW) multiple-input multiple-output (MIMO) [Ins22] radar of the mmWave family produced by Texas Instruments will be used in conjunction with an Optitrack Optical Motion Capture (OMC) system to record participants performing rehabilitation movements and functional activities. The recorded data will then be used to train at least two variations of the PointNet [Qi17a] model: one for single frame processing and another for multi frame processing.

1.2 State of the Art

In the current section, the state of the art of HPE with mmWave radars is described. This exploration starts with a review of published datasets for HPE containing mmWave data. Then different approaches for HPE with mmWave radar pointclouds are discussed. Finally, different classification and regression models designed specifically for point clouds are examined, considering their potential applicability to HPE tasks.

1.2.1 Open-Source Datasets

Since mmWave radars are becoming popular for HPE applications, datasets containing raw radar data or radar pointclouds can be found in the literature. Most of them were recorded in laboratory conditions. They contain a set of movements, usually well-defined rehabilitation movements, where the participants are facing the radar sensors.

The University of Wisconsin-Madison performed a series of HPE studies releasing the mmWave-based Assistive Rehabilitation System for Smart Healthcare (Mars) [An21]. The study was performed with four participants. With the purpose of performing HPE, they created a dataset using a

Texas Instruments IWR1443 Boost mmWave radar to record radar pointclouds at 10 Hz frequency. The 19 reference joints for the human poses were obtained from a Microsoft Kinect V2 sensor at a frequency of 30 Hz. They recorded 10 movements with the participants positioned facing the radar at a distance of 2 m. In addition, they also provide a Convolutional Neural Network (CNN) to predict the joints 3D locations of the person's keypoints.

This paper was followed by the mRI [An22a] paper, also developed by the same group from the University of Wisconsin-Madison. The mRI paper introduced a new dataset containing recordings of 20 participants between the ages of 20 to 28. The participants performed 10 rehabilitation movements in addition to relaxing and stretching free forms and walking. Each recording took approximately one minute. The mRI dataset consists of 5D pointclouds recorded at a frequency of 10 Hz with an IWR1443 Boost mmWave radar. These pointclouds include the spacial coordinates along with the intensity of the signal and Doppler velocity. RGB and depth frames from two Microsoft Kinect V2 sensors recorded at 30Hz frequency, and inertial signals from six wearable Inertial Measurement Unit (IMU) Wit-motion BWT901CL IMU sensors attached to the wrists, knees, head and pelvis of the participant at 50 Hz. The participants performed the movements at 2.4 m from the radar.

The MM-Fi dataset [Yan23b] comprises RGB data, depth frames, Light Detection and Ranging (Lidar) pointclouds, mmWave radar pointclouds and channel state information (CSI) from WiFi. The dataset is annotated with 2D and 3D human keypoints and 27 action categories. The sensors used were an IWR6843 60-64GHz mmWave radar with a frame rate of 10 Hz, the Ouster OS1 32-line LiDAR, and a pair of TP-Link N750 WiFi access points and a Kinect. The participants were recorded at 3 m from the sensors. Since radar pointclouds are sparse and some frames do not contain points, they aggregated the points from consecutive 0.5 seconds to increase the number of points per frame up to 128. The study encompassed 13 daily activities and 12 rehabilitation exercises. They obtained 2D keypoints from the two infrared cameras and used the cameras intrinsic and extrinsic parameters to triangulate the 3D keypoints.

Although a few datasets containing radar data for pose estimation were found in the literature, obtaining the data for mRI proved to be challenging since it was not available on Github. Additionally, the movements of the skeleton target data from MM-Fi were not as fluid as expected, exhibiting some shaking, likely attributable to estimation errors derived from the RGB data. The Mars dataset, although smaller than the other two, provided easy access to the data and showed quality in both radar pointclouds and 3D keypoints.

1.2.2 Human Pose Estimation Methods for Radar Pointclouds

mmWave radars represent an interesting source of information for HPE related tasks. Consequently, mmWave generated pointclouds provide a comprehensive representation of the scene and the objects in it. They are able to accurately calculate distances and velocities, which are important components of human movements. In conjunction with ML methods, mmWave generated pointclouds have proven to be useful to estimate human poses. This section intends to explore the different methodologies followed by researchers to process mmWave radar pointclouds for HPE.

Since their introduction, CNN [Lec89] models have taken a main role in solving image processing task. 2D convolutional layers are specifically designed to process matrix-shaped data. Therefore, they are an accurate choice when it comes to processing data with a strong spatial correlation, like images. Radar pointclouds are not matrix shaped, instead they represent an unordered set of sparse independent points. A CNN cannot be used, unless some prior transformation is performed to convert the pointclouds into a grid-like format.

The University of Wisconsin-Madison in the Mars **p**aper, proposed to order the points within the pointcloud according to their x, y and z values to later rearrange the points into an 8×8 matrix of points. With this setup, each point would be similar to a pixel in an image, facilitating the use of a CNN. The network architecture consisted of two 2D convolutional layers with dropouts followed by two fully connected layers with batch normalization. This rather simple architecture achieved an average Mean Absolute Error (MAE) of 5.87 cm for the 3D keypoints estimation.

Another approach was proposed by mmPose [Sen20]. They projected the 3D points from an mmWave pointcloud into the orthogonal planes xy and xz. Two images were obtained by dividing the planes in 16×16 grids and annotating each grid with features extracted from the points in them. Each of the images is fed to a CNN module with three convolutional layers, 20% dropout and rectified linear unit (ReLU) activations. The CNNs outputs are concatenated to form a forked architecture that ends in three layers Multilayer Perceptron (MLP). The output of the network are the (x, y, z) coordinates of each joint as a flattened vector.

1.2.3 Machine Learning Models for Pointclouds

There are many creative architectures focused on pointclouds. However, not all of them have been tested on radar pointclouds. In this section, methods originally developed for general point clouds are introduced, providing insights into their potential applicability to radar point cloud data.

VoxelNet [Zho17] is based on the idea of dividing the 3D space into equally spaced 3D voxels. A voxel can be understood as the 3D counterpart of a 2D pixel. It represents a small volumetric portion of the space. In a 3D space, a voxel is a regular grid annotated with a value and voxelization is the process of dividing a portion of the 3D space into these regular grids assigning them values according to a function. In the case of pointclouds, voxelization is done by assigning a value to the voxels that have the presence of an element of the pointcloud, similar to the idea of mmPose [Sen20]. Then it annotates each voxel with features extracted with an MLP from the points within the voxel. Finally, 3D convolutions are applied to the voxels for object detection.

PointGNN [Shi20] is a method designed for Lidar pointclouds. They proposed to use a graph representation of a pointcloud and designed a Graph Neural Network (GNN) to detect objects. The graph edges connect a point to its neighbors within a fixed radius. However, since Lidar pointclouds usually have tens of thousand of points, this method includes a downsampling phase to reduce the complexity of the graph. The pipeline consists of creating voxels of dense neighborhood of points and extracting their features using an MLP to downsample the number of vertexes of the graph. The graph is constructed with the downsampled pointcloud. After the graph is constructed, it is processed with a GNN followed by a MLP. It outputs the category and bounding boxes of the objects to which each vertex belongs. It is also able to detect multiple objects in a single shot.

As explained before, there are quite many techniques that aim to extract structural information from pointcloud data. Usually these approaches are either too expensive as using 3D convolutions or need some kind of preprocessing or transformation of the input data like voxelization or rendering. PointNet [Qi17a] is a model designed to perform inference directly on point clouds, leveraging efficiency without the need for preprocessing or input transformation. The model input is represented as a batch containing a set of point's coordinates represented as x, y, z for the 3D coordinates, however, additional features can be added to expand the information provided by the points.

They proposed two main architectures; one for classification and the second for object segmentation. The classification model was trained on the ModelNet40 [Wu15] shape classification benchmark. Although Pointnet was not initially proposed for radar pointclouds, it should be possible to apply the principles under its design to HPE with mmWave radar pointclouds. One of the goals of this thesis is to prove this assumption.

1.3 Research Questions

The appearance of mmWave radars and the possibility to use them indoors for home applications opens new opportunities for HPE applications. mmWave radars provide a vast amount of information about the location and velocities of targets. The scope of this work includes performing HPE on 3D pointclouds obtained from radar data. This work poses the following research questions:

- Can HPE models based on Pointnet architecture accurately estimate human poses from radar pointclouds?
- How do HPE models based on Pointnet architecture perform compared with other approaches described in the literature?
- Does a multiple-frame approach offer significant performance advantages over a single-frame approach?
- Can Pointnet based models be robust against noise and body occlusions?

1.4 Objectives

From the previously mentioned questions the following objectives are derived:

1. Examine the current state of HPE models as documented in existing literature.
2. Develop a dataset for HPE comprising mmWave radar data and human keypoint annotations obtained from an OMC system.
3. Develop and evaluate HPE models based on Pointnet architecture.
4. Compare the outcomes generated by the models designed during this thesis and those reported in existing literature.
5. Assess the performance differences of employing a multiple-frame approach compared to a single-frame approach.
6. Evaluate the robustness of the models designed during this thesis to body occlusions.

Chapter 2

Fundamentals

In this chapter, important topics for the understanding of this thesis will be introduced.

2.1 Machine Learning

"Can Machines think?" Alan Turing posed this question in 1950 [TUR50]. At the time only a few could think about the possibility of a sentient machine, and even today, the answer to that question remains unknown. There are as many accepted definitions of Artificial Intelligence (AI), and the reason is, that there is not an infallible way to define intelligence.

John Haugeland defined AI as:

> "The exciting new effort to make computers think …machines with minds, in the full and literal sense." [Hau85]

And Raymond Kurzweil defined it as:

> "The art of creating machines that perform functions that require intelligence when performed by people." [Kur90]

Both definitions are correct. The first one corresponding to strong AI, which pursuits to create machines whose intelligence is equal to humans and possess consciousness. The second definition corresponds to weak or narrow AI [Sea80]. Weak AI focuses on training machines to perform human tasks with a high success rate. Most of today's AI is powered by weak AI. In practice, AI is a branch of computer science that equips machines, typically computers, to imitate human behavior.

There are many kinds of systems that fall in the category of AI. Such as rule-based and expert systems [Jab15], search algorithms, genetic algorithms [Rus16] and ML algorithms. ML is a subset of AI methods. ML methods distinguish themselves from other AI methods by identifying patterns on data and improve their decision-making without relying on predetermined rules.

ML problems can be classified in two groups: classification and regression problems. Regression problems are solved by predicting continuous values based on the input. Examples of regression problems are predicting stock prices using historical data, estimating house prices base on location and building characteristics [Ann16], forecasting the amount of traffic accidents based on weather conditions. On the other hand, classification problems seek to fit the input into one or more classes. This is typically done following a probabilistic approach and selecting the classes that present higher probability for a given input. Examples of classification problems are determining whether a passenger from the Titanic survived based on their name and physical characteristics [Cuk12], or image classification based on their content [Rus14].

2.1.1 Deep Neural Networks

Deep Neural Networks (DNNs) are arguably the most used ML methods nowadays [LeC15]. DNNs are versatile and extensible, making possible to adapt them to almost any problem. They are extensively used computer vision [Sim15; He15], natural language processing [Tou23] and reinforcement learning [Mni13].

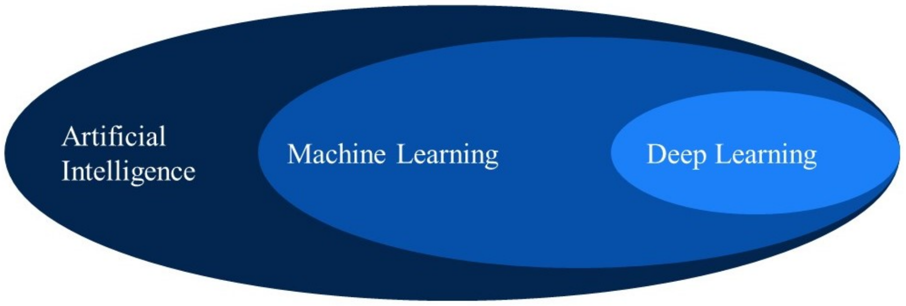


Figure 2.1: Schematic diagram of the relationship between Artificial Intelligence, Machine Learning, Deep Learning

DNNs are based on the Rosenblatt's perceptron [Ros57; Ros58]. The perceptron is inspired on the physical connections of the nervous system. In the nervous system, synaptic connections between neurons are strengthened or weakened based on the frequency and timing of input signals, a process known as synaptic plasticity. This process facilitates changes in the neural structure that corresponds to the learning process. DNNs use the same principle to activate artificial neurons or perceptrons and propagate these activations to other artificial neurons

$$y = \sum_{i=0}^{d-1} w_i x_i + w_d. \tag{2.1}$$

Equation 2.1 calculates the output y of an artificial neuron or perceptron that receives d inputs x_0, x_1 ... x_{d-1}. The perceptron has d weights parameters w_0, w_1 ... w_{d-1} that are multiplied by the inputs influencing the output of the neuron. Additionally, the neuron has a bias parameter w_d, which is added to the product to shift the output.

DNNs are commonly structured as a stack of layers composed by neurons, where each layer transforms the information coming from the previous layer. This kind of DNN is called feedforward network. The propagation of information in feedforward DNNs is made forward, in contrast with the nervous system, where the neurons are connected in all directions.

Fully Connected Layer

Typically, perceptrons are disposed parallelly to form a fully connected layer [Goo16]. This, propitiates that multiple neuron outputs can be calculated from the same set of inputs. Consequently, multiple layers can be stacked together and processing the input of the previous layer and feeding their outputs to the next layer.

Since a fully connected layer, is formed by a set of parallel neurons, the output $\mathbf{y} \in \mathbb{R}^k$ of a layer with k neurons, can be expressed in terms of matrices.

$$\mathbf{y} = \mathbf{W}^\top \mathbf{x} \tag{2.2}$$

The input vector $\mathbf{x} \in \mathbb{R}^{d+1}$ is composed by x_0, x_1 ... x_{d-1} and extended with the value 1 for convenient handling of the bias. The weights' matrix $\mathbf{W} \in \mathbb{R}^{k \times d+1}$ is formed by rows, each row contains the weights of a neuron and the bias.

Convolutional Layer

Convolutional layers are specially tailored for grid like data, like images [LeC89]. The main idea is to implement a convolution operation involving the input of the layer $I \in \mathbb{R}^{W \times H}$ and a two-dimensional kernel $K \in \mathbb{R}^{M \times N}$ [Goo16]. The output $S(i, j)$ is computed:

$$S(i,j) = (I * K)(i,j) = \sum_{m=0}^{M} \sum_{n=0}^{N} I(i-m, j-n) K(m,n). \tag{2.3}$$

When it comes to image processing, convolutional layers are able to capture structures in a pixel's neighborhood due to their two-dimensional kernels. This poses a decisive advantage over fully connected layers. Additionally, the parameters of a convolutional layer are limited to those of the kernel independently of the size of the input. A DNN composed by convolutional layer is called CNN.

Pooling Layers

Pooling layers output the result of a symmetric operation within the rectangular neighborhood of a pixel. There are different kinds of pooling layers, for example, the max pooling layer [Zho88] applies the maximum operation while the average pooling calculates the mean. Pooling helps to achieve a representation that is approximately invariant to small translations of the input [Goo16]. This feature allows more flexibility enabling finding the same feature even when the pooling region is slightly shifted.

Activation Functions

So far, fully connected layers and convolutional layers operate by performing linear operations. Linear operations are insufficient for solving complex problems [Hor91; Cyb89]. Although pooling layers introduce non-linearity to some degree, there are more effective ways. Activation functions apply a non-linear transformation to the input allowing it to learn complex patterns and relationships within the data.

The sigmoid function is commonly used as a DNN activation function. It offers a smooth non-linearization while been bounded between 1 and -1

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{2.4}$$

The ReLU [Fuk69] function returns the input value if it is positive, zero otherwise. It is extensively used due to its computational efficiency

$$\text{ReLu}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

However, dying ReLUs is a phenomenon that occurs when a large gradient updates a weight such that it will not give a positive value for any input. If this happens, the ReLU will never activate and the gradients will never update the weight again. Leaky ReLU [Maa13] is a variant of ReLU that attacks this problem. Leaky ReLUs “leaks” a portion of the activation if it is negative

$$\text{LeakyReLu}(x) = \begin{cases} x, & \text{if } x > 0 \\ a \cdot x, & \text{otherwise,} \end{cases} \tag{2.6}$$

where a is the efficient of leakage and usually a small decimal number. The LeakyReLU activation function was utilized for the models in this thesis.

There are other variants like exponential linear unit (ELU) [Cle16], scaled exponential linear unit (SELU) [Kla17] and Gaussian-error linear unit (GELU).

Loss Functions

Loss functions are used to measure how the outputs or predictions of a DNN $\hat{y}$ deviate from the target values or expected outputs y. Since loss functions take the output of a model as their input, they are, in principle, functions over the model parameters, meaning that changing the model parameters will result in an increase or decrease of the loss or error.

One of the most used loss functions for regression problems is the Mean Squared Error (MSE)

$$\text{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2. \tag{2.7}$$

The MSE is always differentiable and has a single global minimum, making it a widely used choice for regression problems. Consequently, it was selected as the loss function for the models in this thesis.

Parameter Optimization

When a DNN model f process an input x, the input values interact with the model parameters θ of the model and provide an output y. This process is called **forward propagation** [Goo16]. The loss $\mathcal{L}(f(x, \theta), y)$ produced by the networks is computed through a loss function. The gradients of the loss function with respect to the model parameters are computed and propagated backwards through the network using the **back-propagation** algorithm [Rum86; LeC98].

The idea of parameters optimization from DNN models is to reduce the loss by shifting the model parameters by small amounts in the opposite direction of the gradient of the loss function with respect to the parameter. Which theoretically results in the loss decreasing after each iteration. The amount the parameter values are shifted is the product of the loss gradient and an optimizer's hyperparameter η called **learning rate**. The parameters update follows:

$$\theta = \theta - \eta \cdot \nabla_\theta \mathcal{L}(f(x,\theta), y). \tag{2.8}$$

This algorithm is called stochastic gradient descent (SGD) [Bot98].

Adam [Kin17] is another widely used optimizer algorithm. The name Adam derives from adaptive moment estimation. Adam is known for its robustness, efficiency, and effectiveness in training DL models [Sin23]. Consequently, Adam was chosen as the optimizer for the models in this thesis.

Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a kind of DNNs that are specially tailored to process sequences of inputs. They follow the idea of hidden states that act as a memory of previously processed inputs. The hidden state values interact with the current input to give an output also considering the history of the input sequence.

There are two main RNN architectures: Long Short-Term Memory (LSTM) [Hoc97] and Gated Recurrent Unit (GRU) [Cho14].

GRUs, similar to LSTMs networks, excel at capturing long-range dependencies in sequential data. However, GRUs offer the advantage of being computationally more efficient. Research has shown that GRUs achieve comparable performance to LSTMs while requiring fewer parameters [Chu14]. For these reasons the RNN designed for this thesis uses GRUs.

Training and Testing Deep Neural Networks

Initially the data is divided in three sets, one for training, one for validation and the last for testing. The training data is used to update the model weights. The validation data is used during training as an indicator of how the model performs on unseen data. The model parameters are not updated using the validation data. Lastly, after the model is trained, the test data is used to evaluate the model accuracy.

A **mini-batch** is a subset of samples from the data. Mini-batches are used to calculate the models outputs for multiple samples at the same time. The size of the mini-batch is a hyperparameter of the training process and determines how many mini-batches can be formed from each data set.

The process of training a DNN model consist of performing multiple training iterations called **epochs**. The **number of epochs** is a training hyperparameter that defines how many iterations of the training loop are to be done. In each epoch, the training set is divided into mini-batches and each of them is processed by the model. The loss is calculated between the model predictions and expected targets and the gradients are back-propagated. The optimizer algorithm updates the model parameters. After the parameter's update, the model processes the validation data and calculates the mean loss for the validation.

After the training is over, the data is evaluated on the test data, which determines the overall performance of the model.

2.1.2 Pointnet Model

Pointnet is a DL model that supports the assumption of point permutation invariability present in pointclouds [Qi17a]. The output of the network should be invariant to affine transformations such as rotations, translations or point shuffling.

The Pointnet model expects an input of N 3-dimentional points. It uses a transformation network to construct a 3×3 affine transformation matrix and apply this transformation to the input. Then, features are extracted from the transformed input through a MLP. This MLP has shared weights that allows to transform each input feature independently, sustaining the permutation invariability.

At this stage, each point has been transformed into a feature vector of size 64. A second transformation network predicts a new 64×64 transformation matrix for the extracted features. This transformation matrix is constrained to be close to the identity matrix via an auxiliary error function. The features are transformed by the predicted transformation matrix. Subsequently, they are processed by another MLP to extract feature vectors of size 1024.

At this point, features of size 1024 has been extracted from each of the original points in the input. Due to the network design each point has been processed independently with the same set of weights.

At the end they will aggregate each feature vector from the points using the maxpool (Section 2.1.1) function. Since the maxpool is a symmetric function, the permutation invariability assumption continues to be preserved and a single global feature vector of size 1024 is obtained.

From the global feature vector, the paper proposes two possible architectures: one for classification and the second for point segmentation. The models in this thesis are based on the classification architecture.

2.2 Radar

Radar stands for *RAdio Detection And Ranging*. As its name indicates, this technology uses radio waves to locate objects in its effective range. Unlike cameras that depend on an external light, radars operate by emitting Radio Frequency (RF) waves of their own. These waves can penetrate a variety of objects including fog and clouds enabling the radar to operate even in bad weather and absence of light conditions.

2.2.1 Frequency Modulated Continuous Wave Radar

FMCW radar is a specific type of radar sensor that emits continuous transmission power, while changing its operating frequency [Wol]. It can accurately measure the target range and its highly effective measuring the target's relative velocity. The pulse generated by an FMCW radar synthesizer is typically referred to as chirp, a type of signal in which the frequency of the waveform changes over time. The range resolution of the FMCW radar is determined by the bandwidth of the chirp, which is the extent of the frequency change during this period. The components of an FMCW radar are basically transceiver and a control unit with a microprocessor. First the chirp is generated in the Voltage Controlled Oscillator (VCO), then it is transmitted by the transmitter antenna (TX ant.), next the waves bounce on the object and are reflected in various directions. Some of these echo signals are received by the radar's receiver antenna (RX). At the mixing stage, echo signal is compared with the current transmission frequency. Components such as the coupler, amplifiers, and filters play crucial roles in the process by directing the signal, amplifying it, and filtering out unwanted frequencies, respectively.

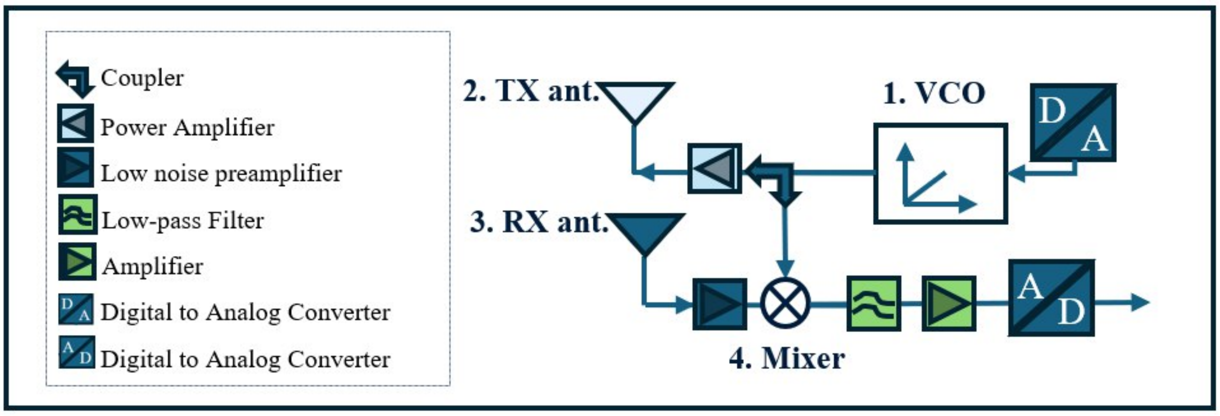


Figure 2.2: Block Diagram of an FMCW radar sensor

2.2.2 Constant False Alarm Rate (CFAR)

An important issue while dealing with radar data is to detect targets in the presence of noise. A basic way to do so is to set a threshold $T(t)$ and ignore all the received signals with an amplitude $A(t) < T(t)$. This threshold regulates the number of targets detected. The higher it is, the fewer targets will be accepted, and at the same time, the number of false alarms is reduced. However, even with a high threshold, noise can still cause false alarms, therefore, an adaptive threshold is used.

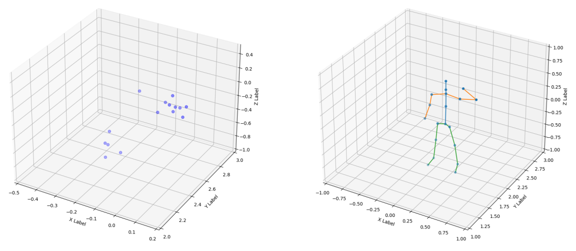
The idea of Constant False Alarm Rate (CFAR) is to have a variable threshold to enforce the false alarm rate to be constant. Cell-Averaging Constant False Alarm Rate (CA-CFAR) consists in dividing the signal into cells and calculating a threshold for each of them. The cell for which the threshold is being calculated is called Cell Under Test (CUT) and its threshold is calculated by averaging the power level of the cells around to the CUT ignoring its immediate adjacent cells. A cell whose amplitude exceeds the average amplitude of the cells around will be acknowledged as a target.

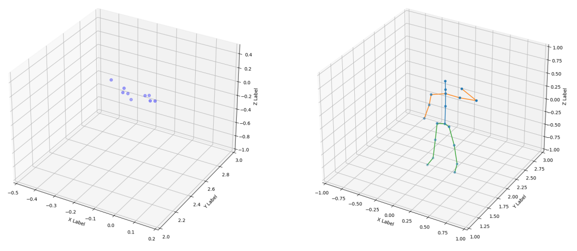
2.2.3 Processing Sparse Radar Pointcloud

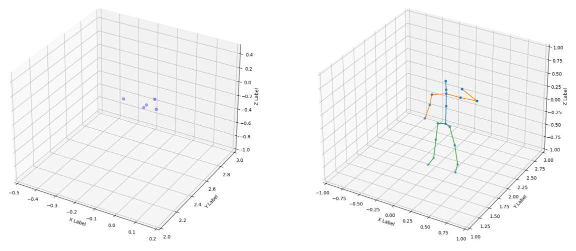
Once the targets are found, they can be represented as a radar pointcloud. Pointclouds from radar are a sparse type of data as shown in figure 2.3. Some frames of pointclouds contain few detections or are empty, causing the ML models to present high errors. Even more interesting is to observe that small changes in the target space (skeleton pose) are correlate with great changes in the input space.

Most ML models require a consistent input shape, not allowing to use a variable number of points as an input. Therefore, the pointclouds need to be standardized to match the model architecture. It is a common practice that radar pointcloud datasets contain a fixed number of points per frame. When a pointcloud has fewer points than necessary, the missing points are filled with zeros. On the other hand if the expected number of points is exceeded, the detections are cropped based on the signal.

Often radars are used in environments where more than one object is present. In this cases semantic segmentation plays a roll in determining which points represent an object instance, object differentiation and removing noise produced by the CFAR. A common approach, used specially in the Automotive industry are clustering algorithms for grouping points into instance candidates. The work proposed by [Sch19] propose a detection filtering to remove points that correspond to static objects, followed by point clustering using DBSCAN algorithm [Est96] and finally a domain


(a) Frame 100


(b) Frame 101


(c) Frame 102

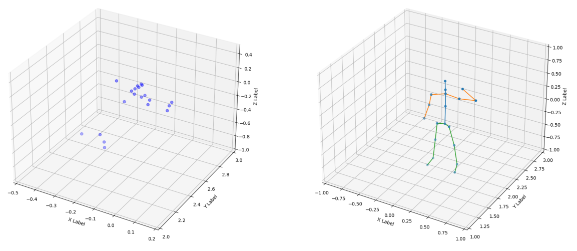

(d) Frame 103

Figure 2.3: Consecutive snapshots of radar poinclouds and their corresponding target human pose.

knowledge based cluster merging to refine the final clusters.

Another way to enhance the information contained in a frame is to fuse many frames into a single one. Each radar pointcloud contains a variable number of points. It is also possible to have frames with few points or even none of them. One of the possible approaches merge the information from consecutive frames to create a single enhanced frame. This was the idea proposed by Fast and Scalable Human Pose Estimation (FUSE)[An22b] and followed by [Yan23b]. The method proposed to fuse the frames is concatenating them.

Be $f[k]$ the k^{th} frame in the dataset. Then the fused frame $F[k]$ is the result of concatenating M consecutive frames as follows:

$$F[k] = \{f[k-M], f[k-(M-1)], \dots, f[k], \dots, f[k+M-1], f[k+M]\}$$

Since they take not only frames from the past but also from the future, it is not suited for real-time applications. However, it can be fixed by only taking a few frames from the past and benefit from the additional information.

Chapter 3

Data Acquisition

A dataset comprising radar generated pointclouds and target skeleton keypoint positions obtained from humans was recorded. The study was performed in the Faculty of Engineering of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), in a room equipped with a OMC system and a mmWave radar.

3.1 Setup

This sections will describe the characteristics of the room where the data was acquired and the sensors and devices involved in the recording process.

3.1.1 Optical Motion Capture

An Optitrack OMC system, comprising 12 cameras strategically positioned around the recording room, was installed for data capture. The room also had a dedicated computer with Motive 2.3.1 installed, which allowed the interaction with the OMC system. Motion capture suits were worn by the participants. These suits allowed reflective markers to be attached to them. Motive provided a number of predetermined marker sets that defined biological landmarks where the reflective markers should be attached to the motion capture suit. When the markers are placed on the body in accordance with the guidelines of a marker set, they will be fitted on a virtual skeleton in Motive. For this thesis, the Conventional (39) [Optb] marker set was selected.

3.1.2 Radar

A radar box equipped with a 60 GHz FMCW MIMO IWR6843AOP mmWave [Ins22] radar was used to obtain the input data for our study. The radar box comprises a mmWave radar and a microcomputer for real time processing. The radar box records radar raw data and simultaneously trigger the OMC.

3.1.3 Room

The recordings were performed in a special room equipped with an Optitrack OMC system. The room dimensions were 10.60 m length, 4.72 m width and 3.27 m height. In the center of the room, a rectangular area of 3.55 m length by 3.05 m width was marked. This area was in the radar field of view, and the OMC cameras were oriented to cover it. The participants were recorded in this area. An additional line was added in the center of the area, for the participant to walk towards the radar and another obliquous line to walk in a $45°$ angle with respect to the radar center. Figure 3.1 represents a top view of the room.

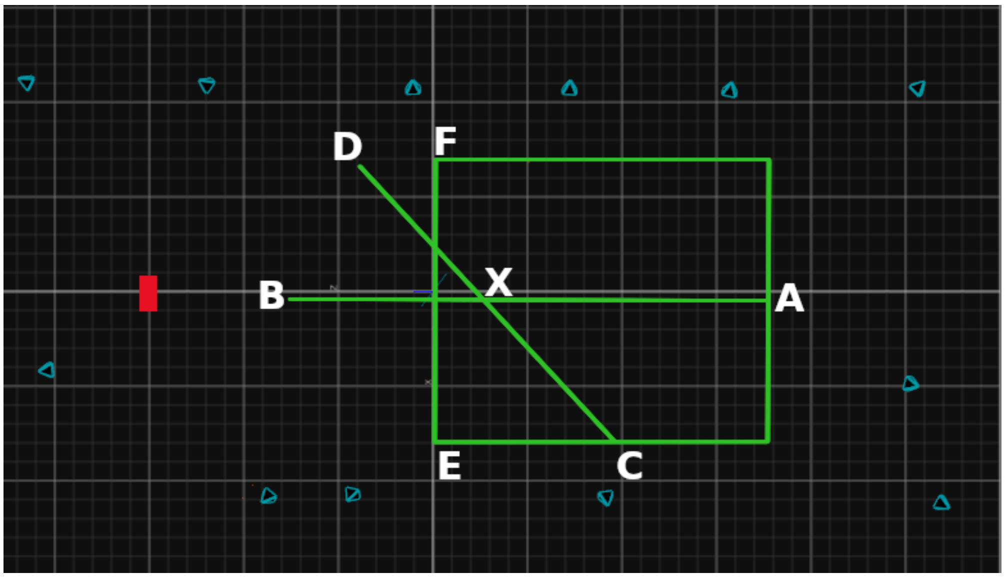


Figure 3.1: Top view representation of the recording room. The camera locations and orientation are represented by blue pyramids. The radar is represented by the red rectangle. The green rectangle represents the area of interest. The **X** marks the location where participant performed the rehabilitation movements. The lines **AB** (0°), **CD** (45°) and **EF** (90°) were used to walk in different directions with respect to the radar.

Before each recording session, the OMC and the radar box were calibrated once using four reference points in the space. These points are created using corner reflectors with reflective markers inside. The corner reflectors are made of three mutually perpendicular, intersecting flat metal surfaces. The material and the shape create a strong echo from the radar emitted wave which is exceptionally detected by the radar. At the same time, the reflective markers inside the corner reflectors are targeted by the OMC. The combination of these corner reflectors and reflective markers allows both systems, the OMC and the radar to find the same 3D coordinates simultaneously. Then, a transformation matrix to transform the radar coordinates system to the OMC coordinates system is obtained.

3.2 Study Procedure

In this section, the study procedure is briefly described. A complete study protocol is presented in the Appendix .1.

3.2.1 Participants

Within this study twelve adult participants were recorded. Four were females and eight males between 25 and 35 years old. At the time of the study, the participants were healthy, and showed no injuries that would prevent them from performing the exercises. At the beginning of each recording session the participants signed a consent form allowing us to use the data obtained during the recordings.

The Table 3.1 summarizes the age, weight and height distributions of all the participants. While tables 3.2.1, 3.2.1 summarize the same information for female and male participants respectively.

Variables	min	max	mean	std
age (years)	25	35	28.92	3.15
weight (kg)	50	93	72.33	12.37
height (m)	168	190	176.25	6.25

Table 3.1: Summary of the age, height, and weight of the twelve participants.

Variables	min	max	mean	std
age (years)	28	35	31.75	2.86
weight (kg)	50	70	62.00	8.15
height (m)	168	173	171.00	2.12

Table 3.2: Summary of the age, height, and weight of the female participants.

Variables	min	max	mean	std
age (years)	25	32	27.50	2.18
weight (kg)	60	93	77.50	10.78
height (m)	170	190	178.88	5.97

Table 3.3: Summary of the age, height, and weight of the male participants.

3.2.2 Data recording

At the beginning of each recording session, the participants were informed about the purpose of the study and how the data will be handled. They consented on giving us the right to use the recorded data. The participants received detailed instructions about each movement they were to perform. They donned motion capture suits, which made it easier to attach the markers. The person conducting the study carefully aligned the markers with the participants' biological landmarks, following the guidelines of the Conventional 39 [Optb] marker set. This marker set, designed for general purposes, includes ample markers for both the upper and lower body, ensuring an accurate representation of the participants' full-body movements

The recording session was divided in four stages. Each stage corresponds to a group of movements: rehabilitation, walking, functional and free movements. Each recording started with the participant performing a T-pose, which helped the OMC to recalibrate the skeleton to the body markers.

There were 10 rehabilitation movements and a T-pose recorded: T-pose (a0), left upper limb extension (a1), right upper limb extension (a2), side upper limb extension (a3), curls (a4), front arm rotation (a5), torso forward bending (a6), left front lunge (a7), right front lunge (a8), squat (a9), side lower limb extension (a10), front lower limb extension (a11).

The rehabilitation movement batch was recorded three times. Each of them corresponded to an action group. Figure 3.2 depicts how the action groups were recorded. The action group 0 was recorded with the participant facing the radar box, the action groups 1 and 2 were recorded with the participant rotated $45°$ and $90°$ counterclockwise, respectively. Recording with the participant facing to different directions is intended to help to evaluate the efficiency of our algorithms to deal with occlusions on radar pointclouds as well as variations in radar resolution across different directions. Each recording in this stage lasted for 36 seconds.

The second batch of movements consisted of four recordings of walking of one minute each. The participant first walked back and forth on a straight line along the radar Y axis (from A to B in Figure 3.1) (w0). Next the participant walked over the $45°$ line marked on the floor (from C to D in Figure 3.1) (w1). On third place, the participant walked on a parallel line to the radar X axis (from E to F in Figure 3.1) (w2). And finally, the participant walked freely on the marked squared area (w3).

The third batch of movements consisted in five functional movements. Simulating picking up and object from the floor (f1), standing up and sitting down (f2), standing up, waking and sitting (f3), standing up, walking around the chair and sitting down (f4) and dancing Macarena (f5). These movements were recorded for 36 seconds each, except for the f5 which lasted for about

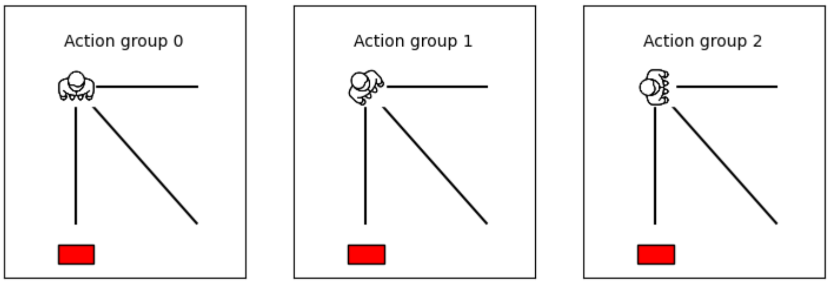


Figure 3.2: Participant's orientation with respect to the radar. The action group 0: participant facing the Radar. The action group 1: participant rotated 45° with respect to the radar. The action group 2: participant rotated 90° with respect to the radar. The radar is represented with the red rectangle.

45 seconds.

And the last batch consisted on a single free movement. The participant shadow-boxed (f6) facing the radar and performing lateral, backwards and forward displacements. The recording lasted for 36 seconds.

3.3 Post-processing

The recorded data was reviewed and post-processed. The recordings sequences were trimmed to remove invalid frames due to the OMC recalibration at the beginning of each recording. The skeleton keypoints were extracted from the OMC data and the radar pointclouds were extracted from the radar raw data.

3.3.1 OMC Data Post-processing

The *.tak* files containing the recordings were saved using the Motive software. These files contained the skeleton model and the marker's location. Each of the markers locations were labeled by the Motive software auto-labeling feature and the skeleton was fitted to the marker's location. However, due to the low frequency of the recordings (15 Hz), the markers were often not labeled or mislabeled resulting in a bad estimation of the skeleton pose. Some markers were lost due to occlusions and some were added due to reflections in the room. These factors required manual post-processing of each recording.

First, all recordings were reviewed to determine its quality based on the number of valid frames it contained. Some recordings were disposed as a result of that.

The files to use were reviewed to determine the start and end frame of each movement, and were trimmed. Each valid frame was reviewed to label unlabeled markers, unlabel markers resulting of noise and relabel those mislabeled. The markers that did not correspond to the skeleton were deemed as noise and removed as a result. Using the edit tools provided by Motive, it was possible to fill the missing markers automatically using cubic interpolation and pattern based interpolation. The process then was repeated until the tracking data reached the best quality. A sequence was considered corrected when it had sufficient markers to make an accurate skeleton estimation.

The recordings were exported as *.vbh* files and the 3D locations of the keypoints were extracted by using bvh-converter. The frames and the keypoints' locations of each recording were saved as a *.npy* file.

3.3.2 mmWave Radar Data Post-processing

The radar datacubes were processed using fast Fourier transform (FFT) [Bri67] and CFAR [Ric14] to obtain radar pointclouds. Static clutter removal was used to remove the static targets. The result were radar pointclouds where each point contained the following seven components: $x, y, z, v, snr, noise, I$, where x, y, z are the 3D coordinates of the detected target, v is the Doppler velocity, snr is the signal-to-noise ratio and I is the intensity.

The 3D coordinates of the pointclouds were transformed to match the OMC coordinates by using the transformation matrix obtained during the calibration process. The pointclouds files were trimmed to match the frames contained in the OMC files.

3.4 mmRadPose Dataset Creation

The post-processing process was very time-consuming. Therefore, only the three action groups of the rehabilitation movement batch were post-processed. The walking, functional and free movements were not included in the current release of the dataset. After the post-processing, some recordings were not usable due to a considerable amount of missing points. Some actions had no valid recordings while others were recorded twice and both recordings were valid. Detailed tables showing how many frames were recorded for each combination of participant, action group and action can be found in the Appendix .2.

The data for the mmRadPose dataset was organized following the structure in Figure 3.3. The first level contains the data from the twelve participants in separate folders named from p1 to p12. Within each participant's folder there are action group folders (actions_0, actions_1, actions_2) that contain the recordings for the three action groups. On the third level, there are action folders numbered from 0 to 11 containing the recordings corresponding to each action.

```
mmRadPose
├── p1                          # participant 1
│   ├── actions_0               # action group 0
│   │   ├── 0                   # T-pose
│   │   ├── 1                   # action 1
│   │   ├── ...
│   │   ├── 10                  # action 10
│   │   └── 11                  # action 11
│   ├── actions_1               # action group 1
│   │   └── ...
│   └── actions_2               # action group 2
│       └── ...
├── p2                          # participant 2
│   └── ...
├ ...                           # participants 3-11
└── p12                         # participant 12
    └── ...
```

Figure 3.3: Dataset folder structure.

The mmRadPose dataset was implemented in python following the python Dataset interface. This implementation allows to select a subset of participants and actions as well as retrieving items with multiple sequential pointclouds enabling time-series processing. When the seq_length hyperparameter is set on the mmRadPose dataset, the items generated would contain a sequence of seq_length radar pointcloud frames as input and a single target skeleton corresponding to the last pointcloud on the input sequence.

For training were selected eight participants (six males and two females). Two participants for validation and two for testing (one male and one female each).

The Table 3.4 makes a comparison between the different datasets available in the literature and ours.

Dataset	# participants	# actions	# sequences	# frames	# 3D keypoints
Mars[An21]	4	10	80	40k	19
mRi[An22a]	20	12	300	160k	17
Hupr[Lee22]	6	3	235	141k	-
mmpose[Sen20]	2	4	-	40k	25
mm-Fi[Yan23b]	40	27	1080	320k	17
mmRadPose	12	30	432	203k	26

Table 3.4: Comparison of the mmRadPose dataset with the datasets in the literature.

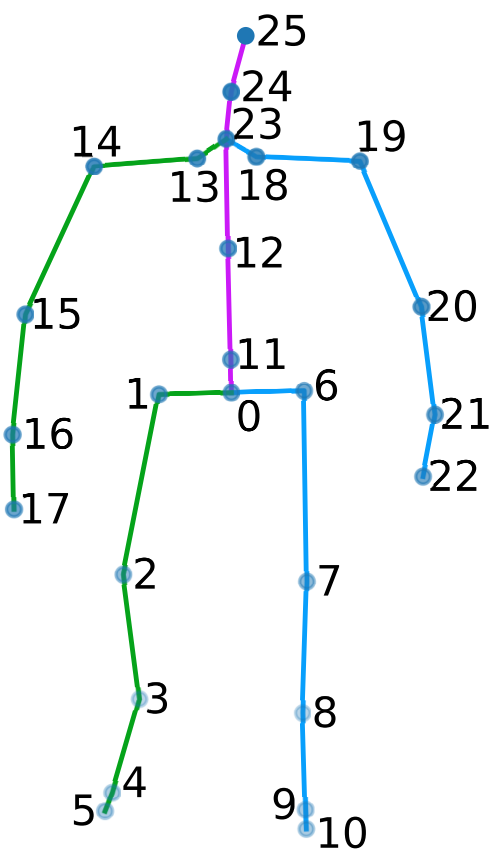


Figure 3.4: Skeleton from the mmRadPose dataset. It consists of 26 keypoints, with the left side colored blue and the right side colored green.

Number	Name	Abbreviation
0	Hips	Hips
1	RightUpLeg	RUL
2	RightLeg	RL
3	RightFoot	RF
4	RightToeBase	RTB
5	RightToeBaseEnd	RTBE
6	LeftUpLeg	LUL
7	LeftLeg	LL
8	LeftFoot	LF
9	LeftToeBase	LTB
10	LeftToeBaseEnd	LTBE
11	Spine	Spine
12	Spine1	Spine1
13	RightShoulder	RS
14	RightArm	RA
15	RightForeArm	RFA
16	RightHand	RH
17	RightHandEnd	RHE
18	LeftShoulder	LS
19	LeftArm	LA
20	LeftForeArm	LFA
21	LeftHand	LH
22	LeftHandEnd	LHE
23	Neck	Neck
24	Head	Head
25	HeadEnd	HeadE

Table 3.5: List of names and abbreviations of the keypoints on the mmRadPose dataset.

Chapter 4

Methods

4.1 Models

For this thesis, two models based on the Pointnet [Qi17a] architecture were designed. The PointnetPose model is a regressor capable of performing HPE with radar pointclouds. Using PointnetPose as a base for extracting features from the pointclouds, PointnetPoseRNN is able to perform HPE using multiple pointcloud frames.

4.1.1 PointnetPose Model

The Pointnet [Qi17a] was specifically designed for pointclouds processing. It was noticed the model can learn a summarized sparse pointcloud which provides an understanding about why it is so robust against missing points and point additions.

PointnetPose is a variation of the original classification Pointnet model architecture (Section 2.1.2). Our proposed architecture is depicted in Figure 4.1. It was implemented using PyTorch following the architecture in the Pointnet paper. However, the input of the Network and the input transform module are modified to accept not only 3D pointclouds, but extended pointclouds including radar extracted features like Doppler velocity, signal intensity and noise. The network contains an input transformation and a feature transformation module that generate transformation matrices to apply affine transformations to the data. First the input is transformed by the input transformation module, then a MLP with shared weights processes each point independently. The resulting tensor is transformed by the feature transformation module and again processed by a shared weight MLP to obtain a feature vector with size 1024 for each of the original input points. The features are condensed by using the max pool operation into a global feature vector. Then a

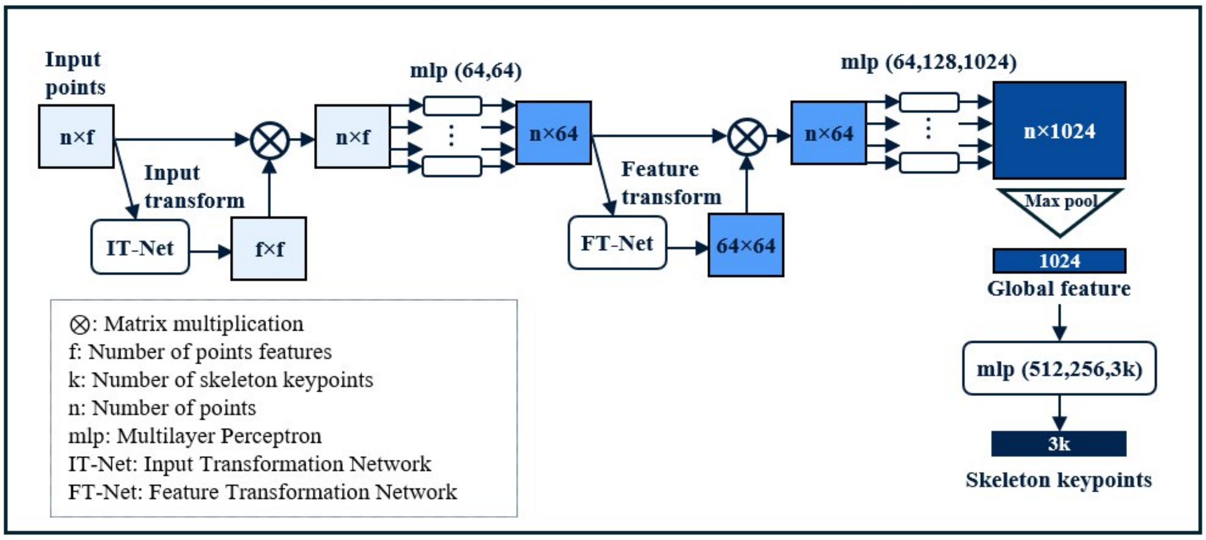


Figure 4.1: PointnetPose architecture.

MLP process the global feature vector to output $k \times 3$ sized tensor containing the predicted 3D skeleton keypoints. k must be set as a network hyperparameter.

4.1.2 PointnetPoseRNN Model

The PointnetPoseRNN model is composed by a feature extractor, GRU cells and a final MLP to generate the 3D skeleton keypoints. The feature extractor comprises the PointNet model without its final layer. The output of the feature extractor connect to a GRU unit with one layer and a hidden state of size 20. The Figure 4.2 depicts the PointnetPoseRNN model.

4.2 Training Procedure

A fully supervised training with the mmRadPose dataset was performed, using mmWave radar pointclouds as input and 26×3D skeleton keypoints as target.

The training loop was implemented using Python. In each experiment the models were trained for 500 epochs using a mini-batch size of 32. The NumPy and PyTorch random generators were seeded using a seed of 42. The Adam optimizer was used with a learning rate of 0.0001.

The Loss chosen for the experiments was MSE loss (Equation 2.7). The model parameters were saved every 20 epochs. Additionally, the models with the smallest validation loss were saved.

The High Performance Computing (HPC) resources were provided by the Erlangen National

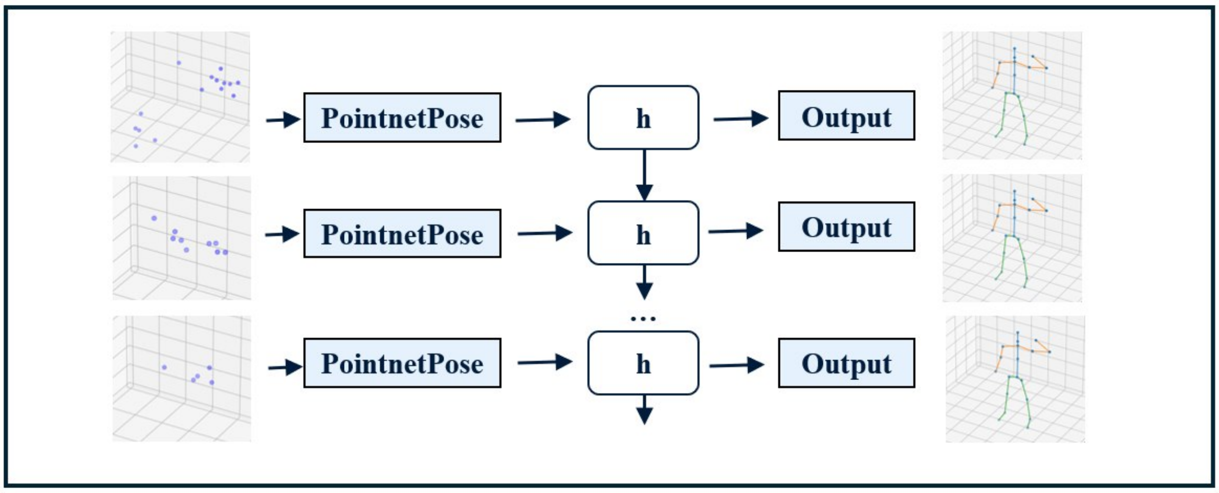


Figure 4.2: PointnetPoseRNN architecture.

High Performance Computing Center (NHR@FAU). An a100 module with 1 x NVIDIA A100-SXM4-40GB GPU node, 2 x AMD EPYC 7713 ”Milan” CPUs with 64 cores each, 1 TB of Memory (DDR4) and 14 TB NVMe SSD was used for training.

4.3 Evaluation Metrics

The models' predicted skeletons' similarity with the target in the dataset were measured using the metrics defined in this section. With S and $\hat{S}$, a target set of skeletons and the predicted set of skeletons respectively, N_S the size of the skeletons sets, N_K the number of keypoints composing each skeleton and $S_{i,k,j}$ the value int the jth axis of the kth keypoint of the ith skeleton in S.

Mean Per Joint Position Error (MPJPE)

The MPJPE calculates the mean over all the euclidean distances between the predicted skeleton keypoints and the target skeletons keypoints

$$\text{MPJPE}(S_i, \hat{S}_i) = \frac{1}{N_K} \sum_{k=1}^{N_K} ||S_{i,k} - \hat{S}_{i,k}||_2 \tag{4.1}$$

$$\text{MPJPE}(S, \hat{S}) = \frac{1}{N_S} \sum_{i=1}^{N_S} \text{MPJPE}(S_i, \hat{S}_i). \tag{4.2}$$

4.3.1 Percentage of Correct Keypoints (PCK)

The Percentage of Correct Keypoints (PCK) is used to predict the ratio of correct keypoint detections. A keypoint detection is considered correct is the distance with the target keypoint location is smaller than a threshold t.

$$C(S_{i,k}, \hat{S}_{i,k}) = \begin{cases} 1, & \text{if } ||S_{i,k} - \hat{S}_{i,k}||_2 < t \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

$$\text{PCK}(S_i, \hat{S}_i) = \frac{1}{N_K} \sum_{k=1}^{N_K} C(S_{i,k}, \hat{S}_{i,k}) \times 100 \tag{4.4}$$

$$\text{PCK}(S, \hat{S}) = \frac{1}{N_S} \sum_{i=1}^{N_S} \text{PCK}(S_i, \hat{S}_i) \tag{4.5}$$

4.3.2 Mean Absolute Error (MAE)

The MAE was used to measure the displacement of the skeleton keypoints in a given axis. This allows to visualize which combinations of keypoint and axis are more affected in the predictions.

$$\text{MAE}_{k,j}(S, \hat{S}) = \frac{1}{N_S} \sum_{i=1}^{N_S} |S_{i,k,j} - \hat{S}_{i,k,j}| \tag{4.6}$$

The MAE on a single axis can also be calculated as:

$$\text{MAE}_{j}(S, \hat{S}) = \frac{1}{N_K} \sum_{k=1}^{N_K} \text{MAE}_{k,j}(S, \hat{S}) \tag{4.7}$$

4.4 Experiments

The experiments were designed to train our models and measure their accuracy on HPE with the mmRadPose dataset. First the PointnetPose model was validated using the data from Mars. Later the model was trained for the three action groups independently.

The validation of the PointnetPose model was done by comparing its results with those presented by the Mars model using the data in the same paper. Both algorithms, Mars and PointnetPose were trained using the data split proposed by the paper. For both models, the MSE loss was used during training for 500 epochs using a mini-batch size of 32. Both loss curves were compared and the metrics with the test set were calculated.

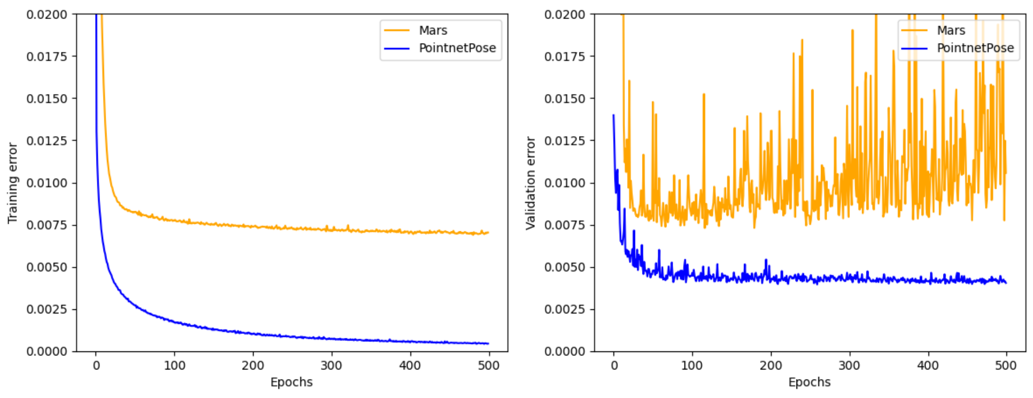


Figure 4.3: Training and Validation loss curves for PointnetPose and Mars models trained on the Mars dataset.

Figure 4.3 shows the training and validation losses for both algorithms. During training PointnetPose and Mars achieved a minimum loss of 0.00043 and 0.00688 respectively. While during validation PointnetPose achieved a minimum loss of 0.00395 upon convergence and Mars achieved 0.00729 without showing any convergence.

The action group 0 contains the recordings, made with the participants facing the radar. This setup is similar to those found in the literature that have achieved a good performance. One of the reasons is that it contains fewer occlusions that the other two action groups, with respect to the radar point of view.

PointnetPose and PointnetPoseRNN were trained using all the actions in the action group 0, including the T-pose. The objective was to observe if PointnetPoseRNN would perform better that PointnetPose. Later the experiment was repeated excluding the T-pose. The reason was that the T-pose is a static movement. Given that the radar pointclouds were processed using the static clutter removal, this movement resulted in few or no points. Which cause the networks to predict poses close to the T-pose when there were few points present.

The mmRadPose dataset recorded seven different point features. It includes the 3D point coordinates, the Doppler velocity of the target, the signal-to-noise ratio, the noise and the signal intensity. Datasets in the literature do not include the noise, and signal-to-noise ratio [An21]. The test is intended to unveil if training with and without this feature would make a substantial difference.

The action group 1 contains the recordings where the participant is rotated $45°$ with respect to the radar. This suggests that the number of limb occlusions should be greater than the action group 1. The recorded participants in the action group 2 are rotated $90°$ with respect to the radar. In this case, half of the body is occluded mos of the time. PointnetPoseRNN was trained on action group 1 and 2 independently.

Lastly, PointnetPoseRNN was trained using all the action groups.

Experiment	Model	Dataset	action group	T-pose	noise	sequence length
0	Mars	Mars	-	-	-	-
1	PointnetPose	Mars	-	-	-	-
2	PointnetPose	mmRadPose	0	yes	yes	-
3	PointnetPose	mmRadPose	0	yes	no	-
6	PointnetPose	mmRadPose	0	no	yes	-
7	PointnetPose	mmRadPose	0	no	no	-
5	PointnetPoseRNN	mmRadPose	0	yes	yes	4
8	PointnetPoseRNN	mmRadPose	0	no	yes	4
9	PointnetPoseRNN	mmRadPose	1	no	yes	4
10	PointnetPoseRNN	mmRadPose	2	no	yes	4
11	PointnetPoseRNN	mmRadPose	0	no	yes	2
12	PointnetPoseRNN	mmRadPose	0	no	yes	10
13	PointnetPoseRNN	mmRadPose	1	no	yes	10
14	PointnetPoseRNN	mmRadPose	2	no	yes	10
15	PointnetPoseRNN	mmRadPose	(0,1,2)	no	yes	10

Table 4.1: Experiments

Chapter 5

Results

This chapter presents the results of training PointnetPose and PointnetPoseRNN under different conditions.

PointnetPose was validated by comparing its performance with that of the Mars model. Experiments were conducted to determine the impact of training using the T-pose recordings. The mmRadPose pointclouds present the values of the noise and signal-to-noise ratio, and the influence of training with and without those values is presented. Additionally, the performance of PointnetPose and PointnetPoseRNN is compared using the same dataset. Finally, the performance of PointnetPoseRNN trained on the three action groups is presented.

5.1 PointnetPose validation

The Mars [An21] paper proposed a CNN model able to process 5-dimentional radar pointclouds. Each point P_i is formed by the features x_i, y_i, z_i, D_i, I_i where the first three components are the 3D coordinates of the point, D_i is the Doppler velocity and I_i is the intensity of the signal. The Mars model and the PointnetPose model were trained for 500 epochs and tested on the test set.

Figure 5.1 shows the PCK metric with ten evenly spaced thresholds ranging from 2 and $20\,\text{cm}$. PointnetPose presented a higher percentage of correct keypoints for all the proposed thresholds. Reaching a $91.17\ \%$ of predictions within $16\,\text{cm}$ distance from the real keypoint position. While Mars reached only $77.48\ \%$ of the predictions correctly classified within the same threshold distance. The MPJPE for all the keypoints was $11.85\,\text{cm}$ for Mars and $7.26\,\text{cm}$ for PointnetPose.

The graphic in Figure 5.2 illustrates the mean position error for each of the skeleton keypoints on the Mars dataset. The shapes of both polygons are similar which suggest that the models have similar strengths and weaknesses while predicting keypoints, however in a different scale. The

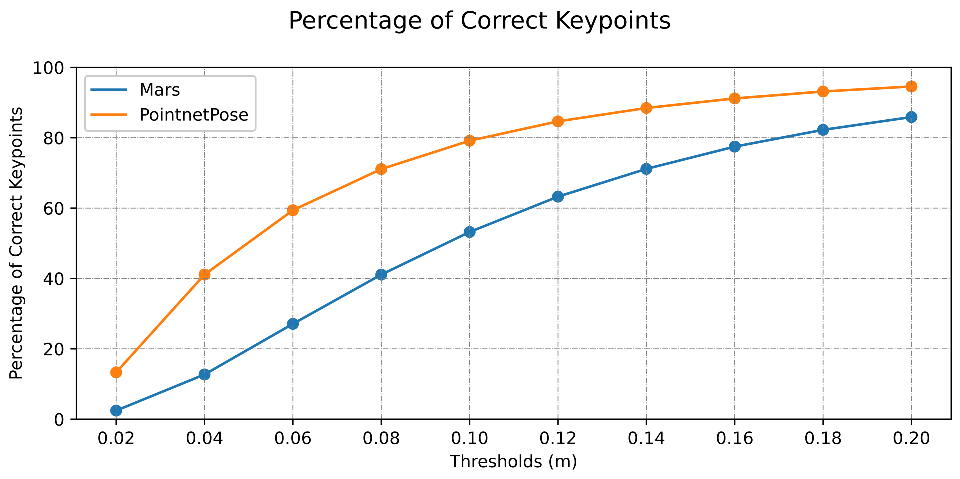


Figure 5.1: Percentage of Correct Keypoints (PCK) calculated for the Mars and PointnetPose algorithms trained on the Mars dataset across ten thresholds ranging from 2 to 20 cm.

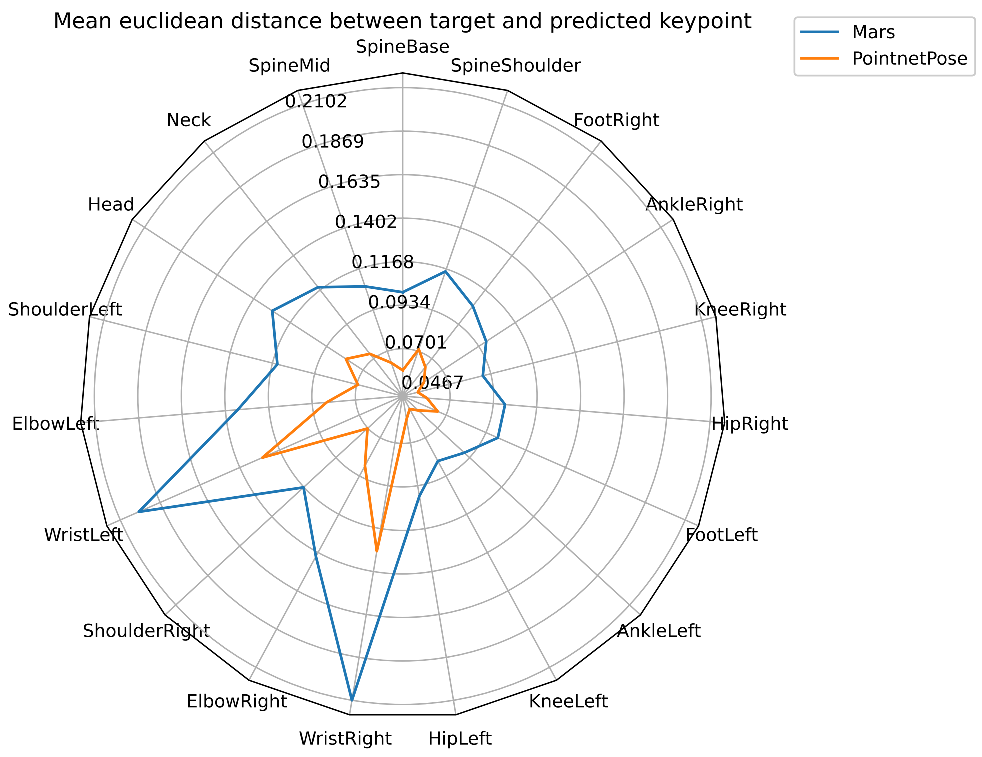


Figure 5.2: The radial axes depict the mean euclidean distance between target and predicted keypoints. Comparison between the Mars and PointnetPose algorithms trained on the Mars dataset.

PointnetPose model surpasses the Mars model in terms closing the distance between the predicted and target keypoints. The highest mean distance is present on the wrists for both models.

5.2 Preliminary Experiments

In this section, the results PointnetPose and PointnetPoseRNN models trained on the mmRadPose dataset will be shown.

Four PointnetPose model were trained on the action group 0 of the mmRadPose dataset. Two of the models excluded the T-pose recordings from the training data, while the other two trained including the T-pose recordings. From the four models, two used all the pointclouds features (3D location, Doppler velocity, Intensity, noise and signal-to-noise ratio), while the other two excluded the noise and the signal-to-noise ratio. The PCK for four PointnetPose algorithms were calculated using ten thresholds ranging from 2 to 20 cm (Table 5.1).

PCK (%) / Experiment	t=0.02 m	t=0.04 m	t=0.06 m	t=0.08 m	t=0.10 m	t=0.12 m	t=0.14 m	t=0.16 m	t=0.18 m	t=0.20 m
2 (T-pose and all vars)	0.61	4.41	13.01	25.18	38.90	52.72	65.26	74.92	81.18	85.02
3 (T-pose and var subset)	0.60	4.03	12.03	23.52	36.70	50.35	62.58	72.54	79.39	83.56
6 (No T-pose and all vars)	0.62	4.71	14.33	28.13	42.85	57.01	69.49	78.44	83.44	86.49
7 (No T-pose and var subset)	0.43	3.18	9.98	20.82	33.57	47.30	60.39	70.91	78.44	83.12

Table 5.1: Percentage of Correct Keypoints (PCK) for PointnetPose models on mmRadPose action group 0.

The MPJPE values for the PointnetPose models are described in the Table 5.2.

Experiment	2 (T-pose and all vars)	3 (T-pose and var subset)	6 (No T-pose and all vars)	7 (No T-pose and var subset)
MPJPE	0.1418	0.1477	0.1362	0.1511

Table 5.2: Mean Per Joint Position Error (MPJPE) for the PointnetPose models.

Both algorithms including all pointcloud features presented a higher PCK that those using a subset of features. From the two models that used all pointcloud features, the one excluding the T-pose recordings outperformed the model using the T-pose. Similarly, from the two models that used a subset of pointcloud features the one excluding the T-pose recordings outperformed the model using the T-pose. In general the best model, according to the PCK curves was trained with all the pointcloud features and excluded the T-pose from the dataset. The model including the T-pose and using a subset of the pointcloud features was the worst performing according to the results of this experiment.

Four models were trained on the action group 0 from the mmRadPose dataset. The training sets used all pointcloud features and excluded the T-pose recordings. On this section, the results of one

PointnetPose model and three PointnetPoseRNN models will be presented. The PointnetPoseRNN were trained using sequence lengths of 2, 4 and 10 respectively.

PCK (%) / Experiment	t=0.02 m	t=0.04 m	t=0.06 m	t=0.08 m	t=0.10 m	t=0.12 m	t=0.14 m	t=0.16 m	t=0.18 m	t=0.20 m
6 (PointnetPose)	0.62	4.71	14.33	28.13	42.85	57.01	69.49	78.44	83.44	86.49
11 (PointnetPoseRNN seq_lenght = 2)	0.88	5.66	15.24	28.52	42.91	56.37	68.13	77.47	83.70	87.22
8 (PointnetPoseRNN seq_length = 4)	0.74	5.21	15.14	29.80	45.74	60.05	71.81	80.57	86.11	89.33
12 (PointnetPoseRNN seq_lenght = 10)	0.79	5.68	16.73	32.23	48.06	61.60	72.81	81.55	87.33	90.47

Table 5.3: Percentage of Correct Keypoints (PCK) for PointnetPose models on mmRadPose action group 0.

Experiment	6 (PointnetPose)	11 (PointnetPoseRNN seq_lenght = 2)	8 (PointnetPoseRNN seq_length = 4)	12 (PointnetPoseRNN seq_lenght = 10)
MPJPE	0.1362	0.1329	0.1247	0.1197

Table 5.4: Mean Per Joint Position Error (MPJPE) for the PointnetPose models.

The results consistently showed a relation between the sequence length and the algorithm performance. However, these improvements do not seem to be quantitatively significant.

5.3 Evaluation of PointnetPoseRNN on mmRadPose

In this section the results of training PointnetPoseRNN using sequence length of 10 in the different action groups will be presented. Three PointnetPoseRNN models were trained independently on each action group of the mmRadPose dataset. A fourth model was trained on the entire dataset.

For each model, the MAE per keypoint and axis combination will be presented as well as the MAE per axis. In order to achieve a deeper understanding of the about the performance of the model, the distances between each predicted keypoint and the target keypoints were computed and the distribution of the distances presented in a boxplot.

Information about the PCK of each model and model and the MAE was presented.

In this section, the performance of the models will be broken down into each action from the action groups. The mean euclidean distance between the predicted keypoints and the target keypoints of the skeletons in the same action and action group will be represented using radar or spider plots. The radial axes of the radar plots corresponds to each of the skeleton keypoints. The names of the skeleton keypoints were abbreviated. Refer to Table 3.5 for a list of the names and abbreviations.

5.3.1 Evaluation of PointnetPoseRNN on the action group 0

A PointnetPoseRNN model was trained using the action group 0. The model achieved a MPJPE of 11.96 cm

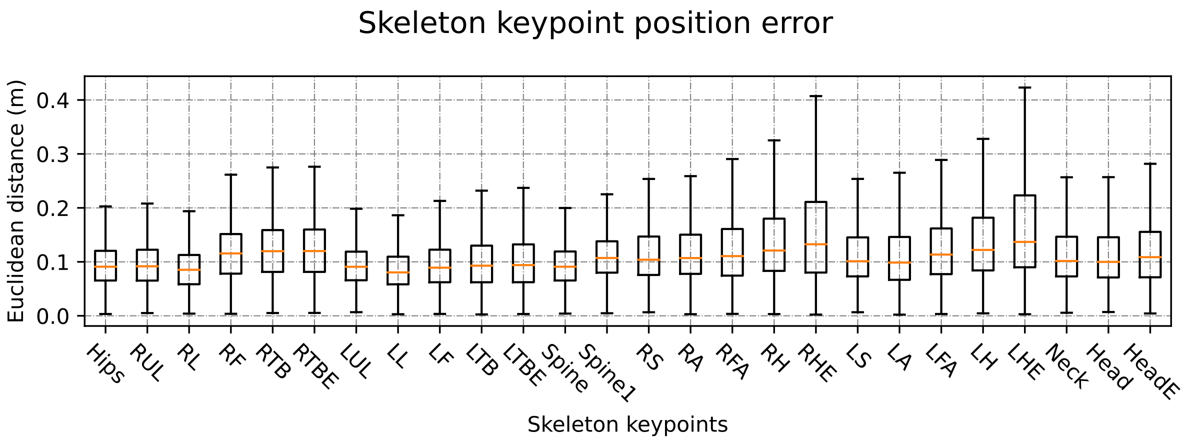


Figure 5.3: Distribution of distances between the target and predicted skeleton keypoints. The predictions were calculated with a PointnetPoseRNN model trained on the action group 0. The outlayers were removed from the graphic.

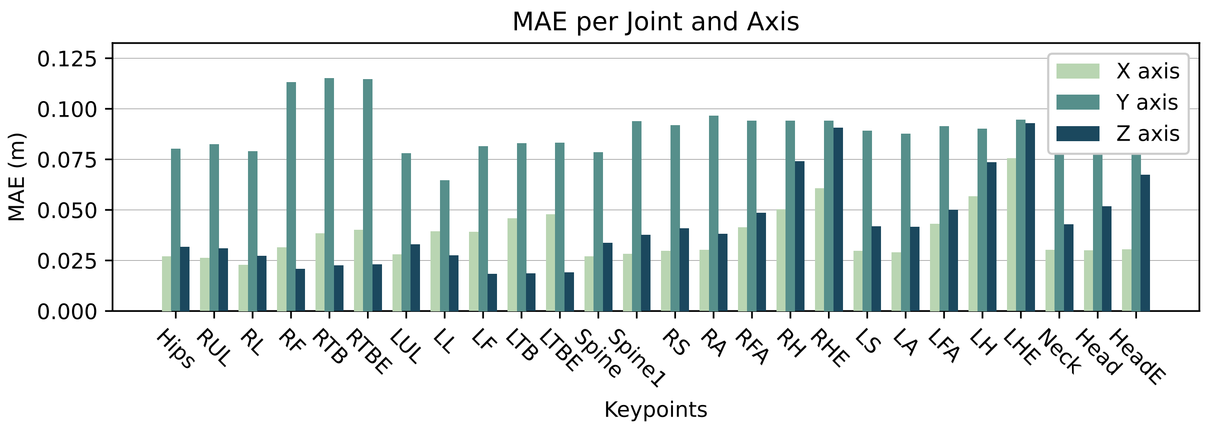


Figure 5.4: Mean Absolute Error (MAE) for each keypoint and axis combination. The Mean Absolute Error was calculated from the predictions of a PointnetPoseRNN model trained on the action group 0 of the mmRadPose dataset.

Thresholds	t=0.02	t=0.04	t=0.06	t=0.08	t=0.1	t=0.12	t=0.14	t=0.16	t=0.18	t=0.2
PCK	0.79	5.68	16.73	32.23	48.06	61.60	72.81	81.55	87.33	90.47

Table 5.5: Percent of Correct Keypoints from a PointnetPoseRNN model trained on the action group 0.

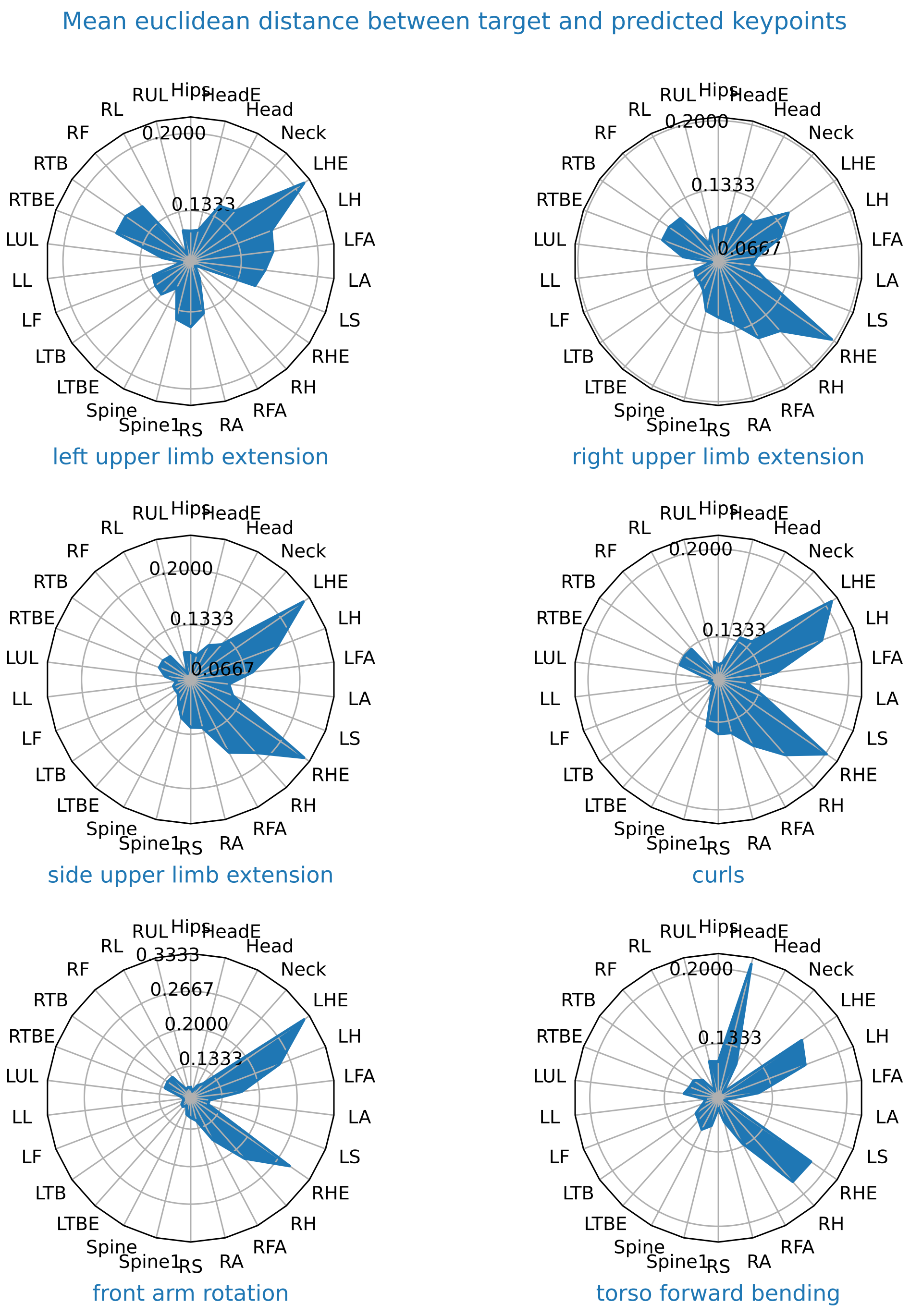


Figure 5.5: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the action group 0 of the mmRadPose dataset. Part 1.

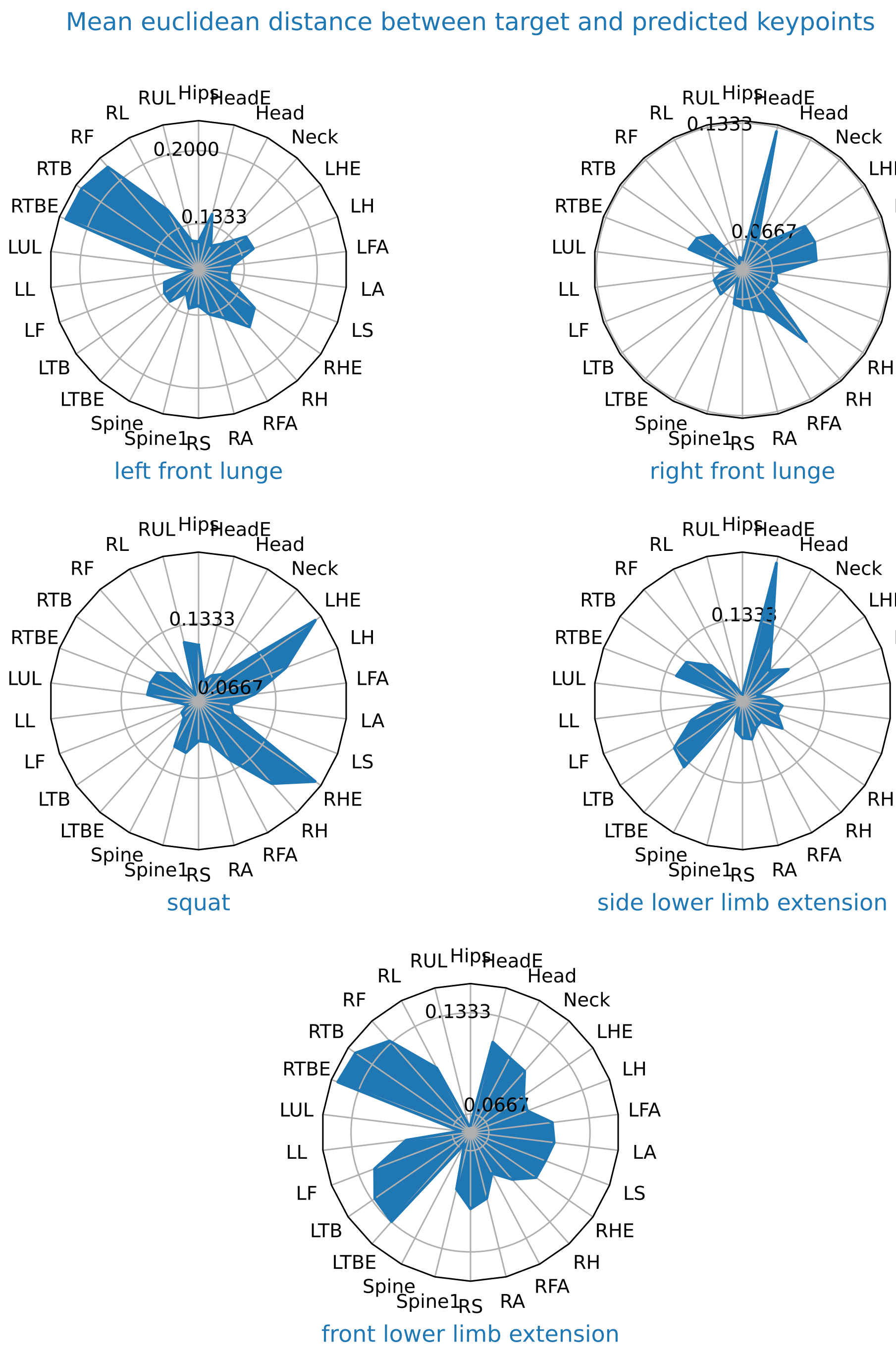


Figure 5.6: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the action group 0 of the mmRadPose dataset. Part 2.

5.3.2 Evaluation of PointnetPoseRNN on the action group 1

A PointnetPoseRNN model was trained using the action group 1. The model achieved a MPJPE of 8.82 cm

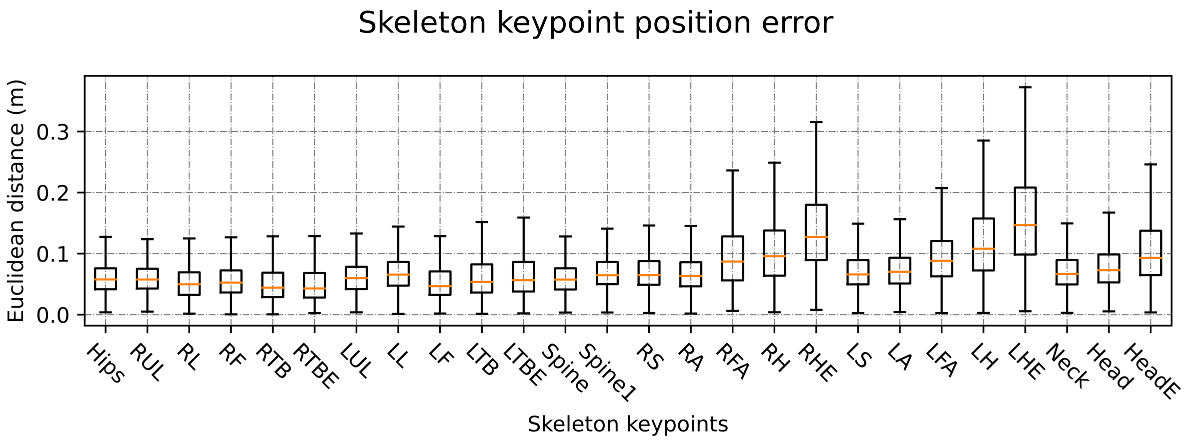


Figure 5.7: Distribution of distances between the target and predicted skeleton keypoints. The predictions were calculated with a PointnetPoseRNN model trained on the action group 1.

Thresholds	t=0.02	t=0.04	t=0.06	t=0.08	t=0.1	t=0.12	t=0.14	t=0.16	t=0.18	t=0.2
PCK	3.45	19.04	42.46	62.32	74.38	82.15	87.31	90.61	92.85	94.34

Table 5.6: Percent of Correct Keypoints from a PointnetPoseRNN model trained on the action group 1.

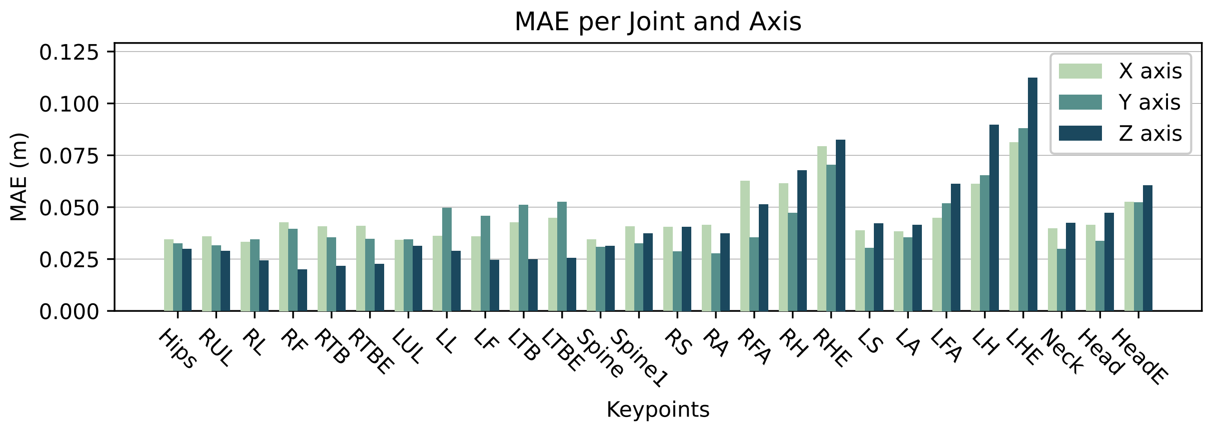


Figure 5.8: Mean Absolute Error (MAE) for each keypoint and axis combination. The Mean Absolute Error was calculated from the predictions of a PointnetPoseRNN model trained on the action group 1 of the mmRadPose dataset.

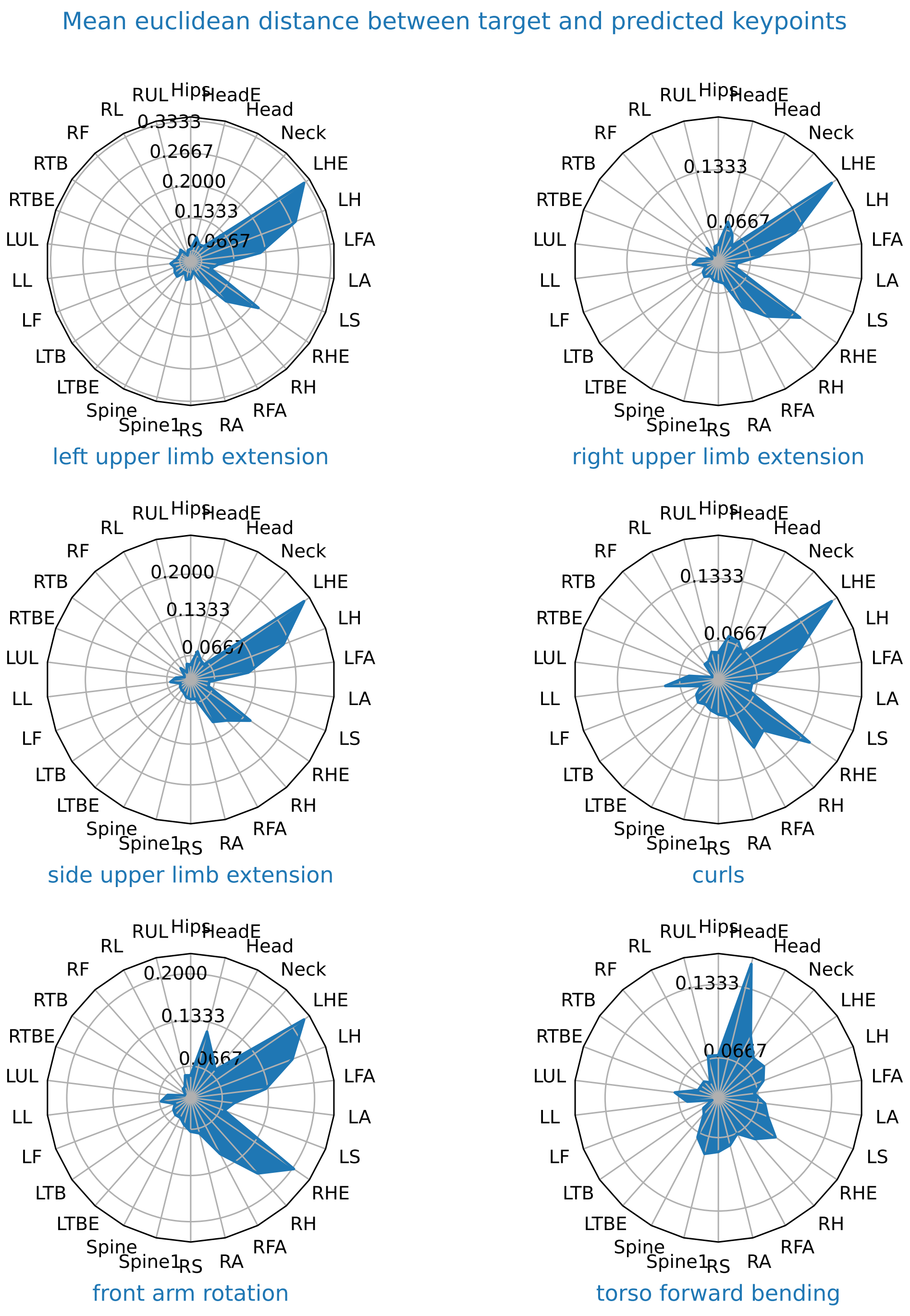


Figure 5.9: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the action group 1 of the mmRadPose dataset. Part 1.

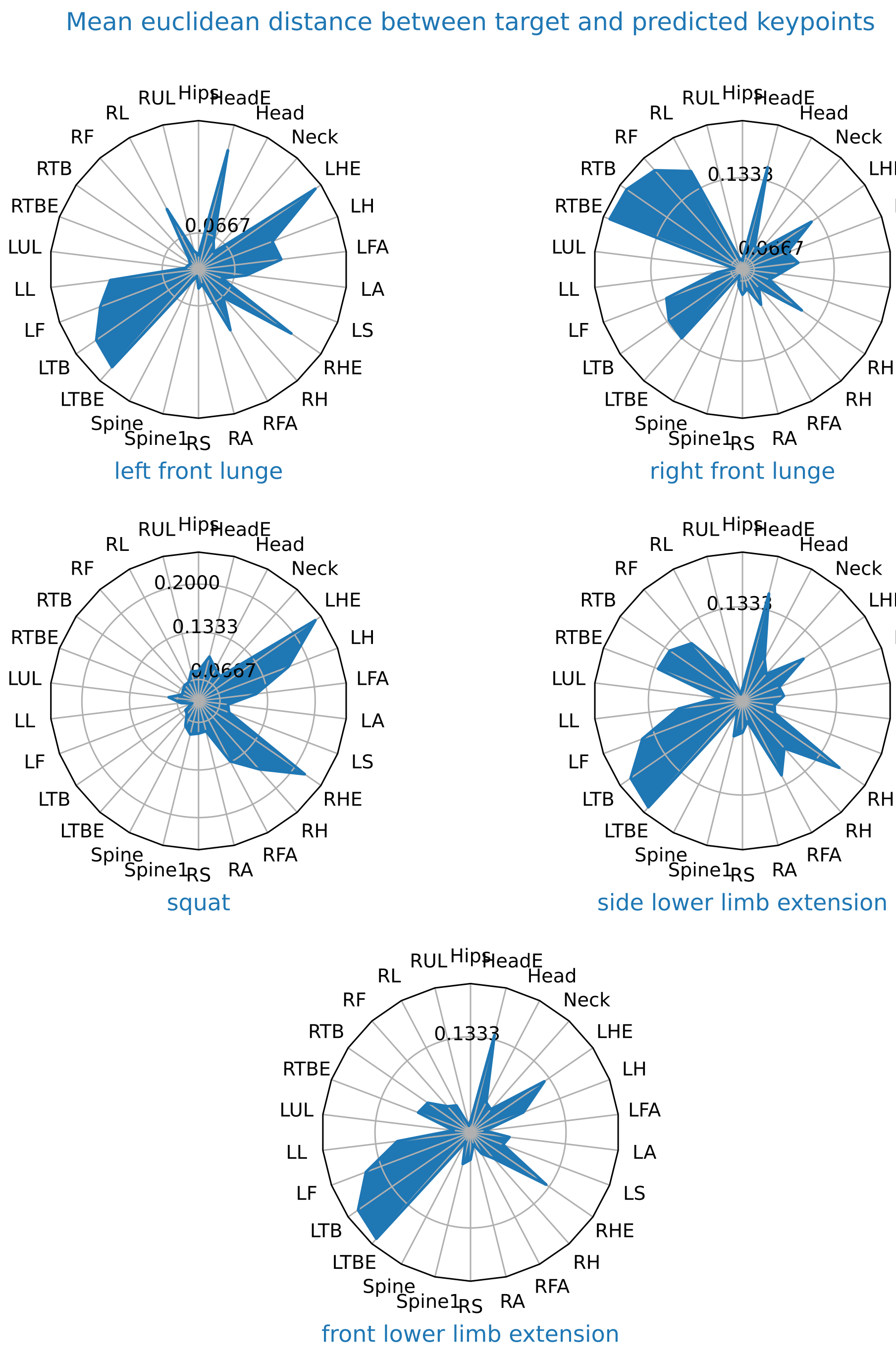


Figure 5.10: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the action group 1 of the mmRadPose dataset. Part 2.

5.3.3 Evaluation of PointnetPoseRNN on the action group 2

A PointnetPoseRNN model was trained using the action group 2. The model achieved a MPJPE of $8.83\,\mathrm{cm}$

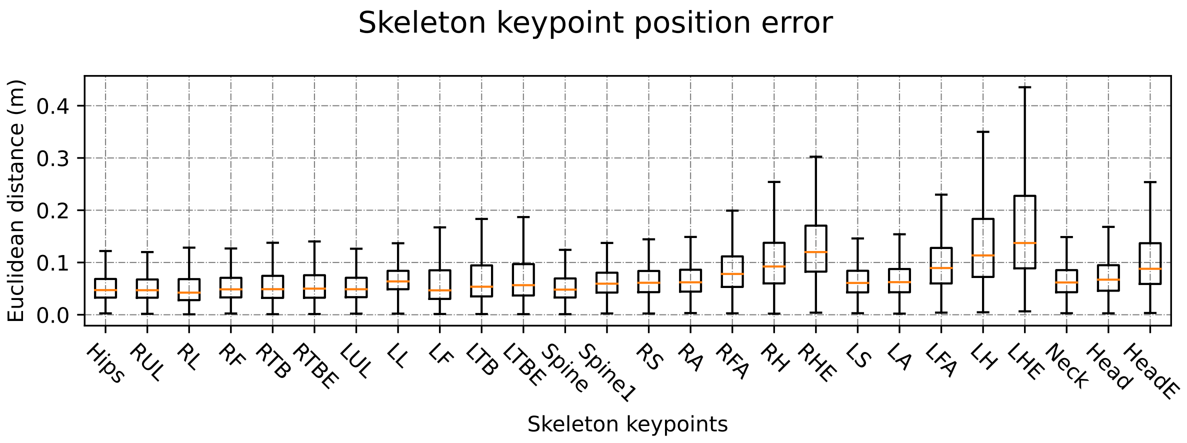


Figure 5.11: Distribution of distances between the target and predicted skeleton keypoints. The predictions were calculated with a PointnetPoseRNN model trained on the action group 2.

Thresholds	t=0.02	t=0.04	t=0.06	t=0.08	t=0.1	t=0.12	t=0.14	t=0.16	t=0.18	t=0.2
PCK	4.67	23.86	46.90	64.23	75.20	82.03	86.66	89.76	91.84	93.33

Table 5.7: Percent of Correct Keypoints from a PointnetPoseRNN model trained on the action group 2.

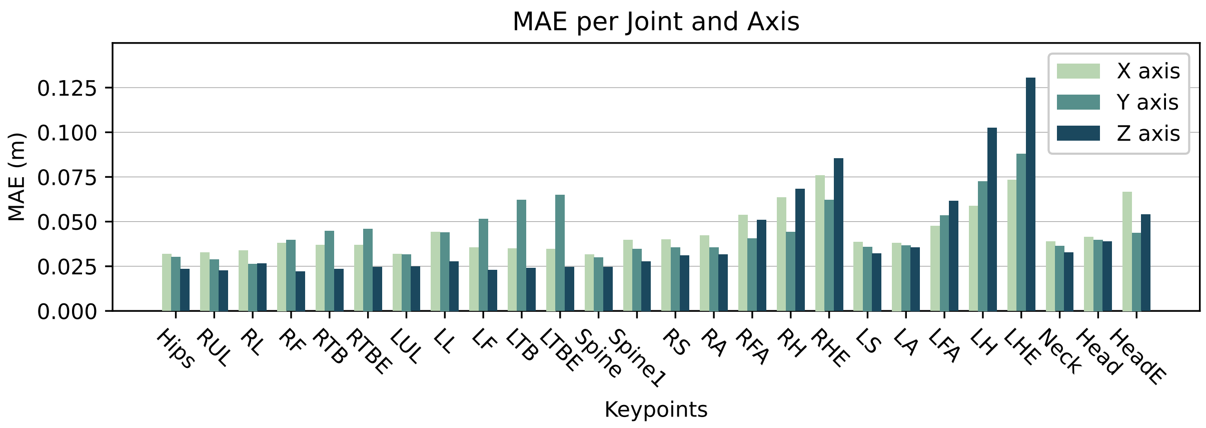


Figure 5.12: Mean Absolute Error (MAE) for each keypoint and axis combination. The Mean Absolute Error was calculated from the predictions of a PointnetPoseRNN model trained on the action group 2 of the mmRadPose dataset.

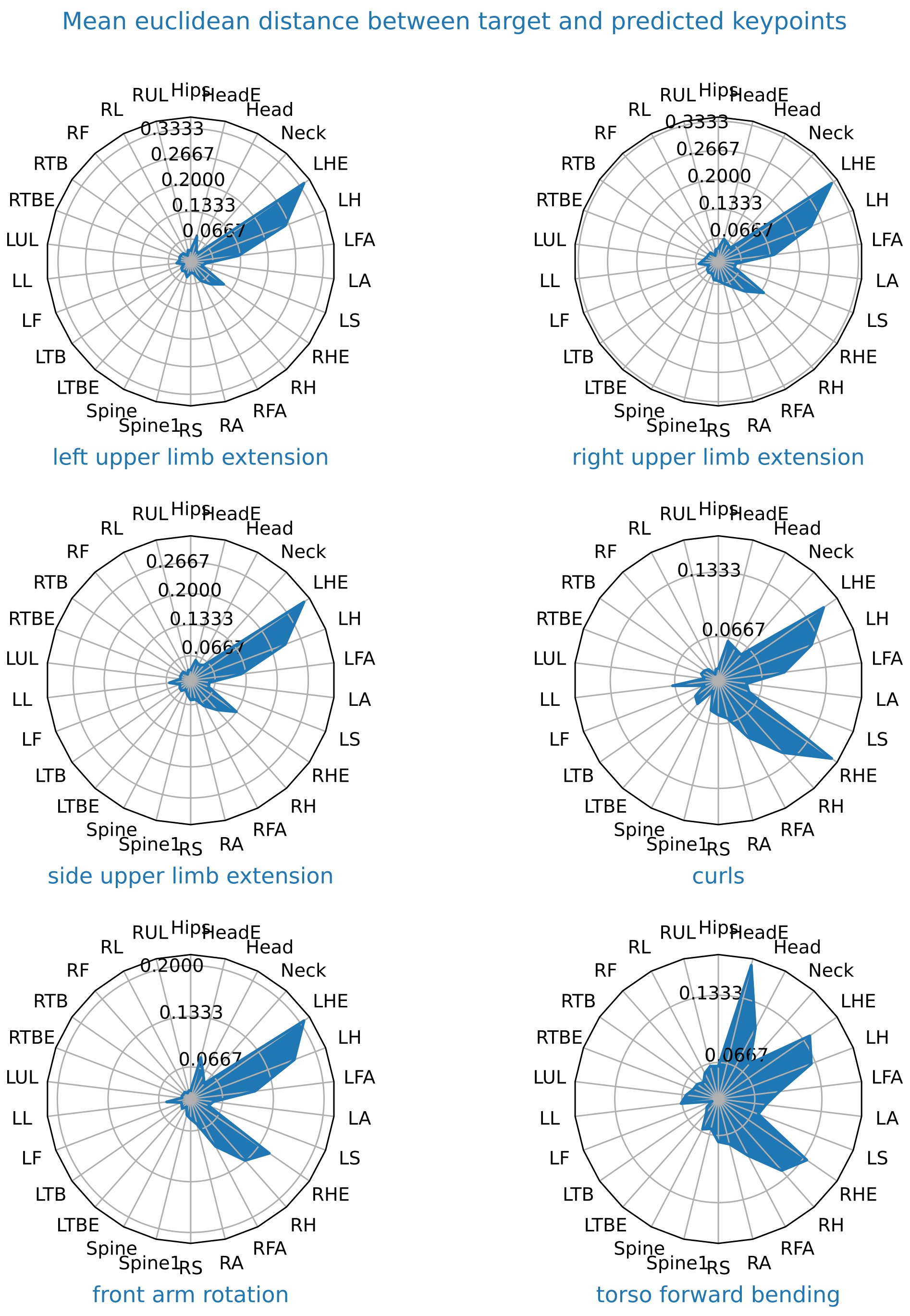


Figure 5.13: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the action group 2 of the mmRadPose dataset. Part 1.

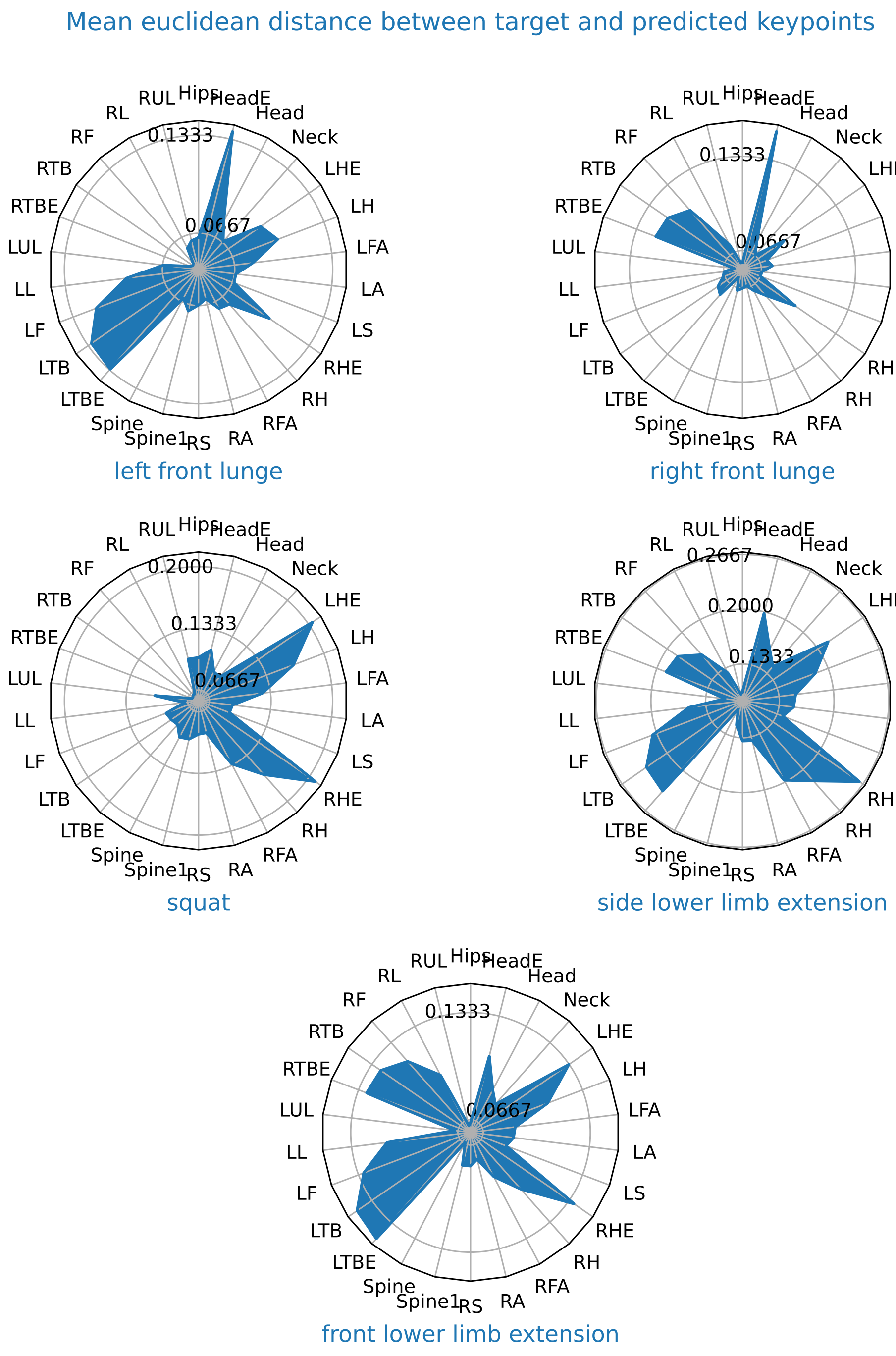


Figure 5.14: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the action group 2 of the mmRadPose dataset. Part 2.

5.3.4 Evaluation of PointnetPoseRNN on all the action groups (0, 1, 2)

A PointnetPoseRNN model was trained using the entire mmRadPose dataset. The model achieved a MPJPE of 9.94 cm. This model could only be trained for 320 epochs in contrast with the other models that were trained for 500 epochs.

This model exhibit no significant performance differences compared with the other models. It was able to achieve a PCK of 67.17 % at a threshold of 10 cm which is even higher than the PCK of the model trained on the action group 0 at the same threshold.

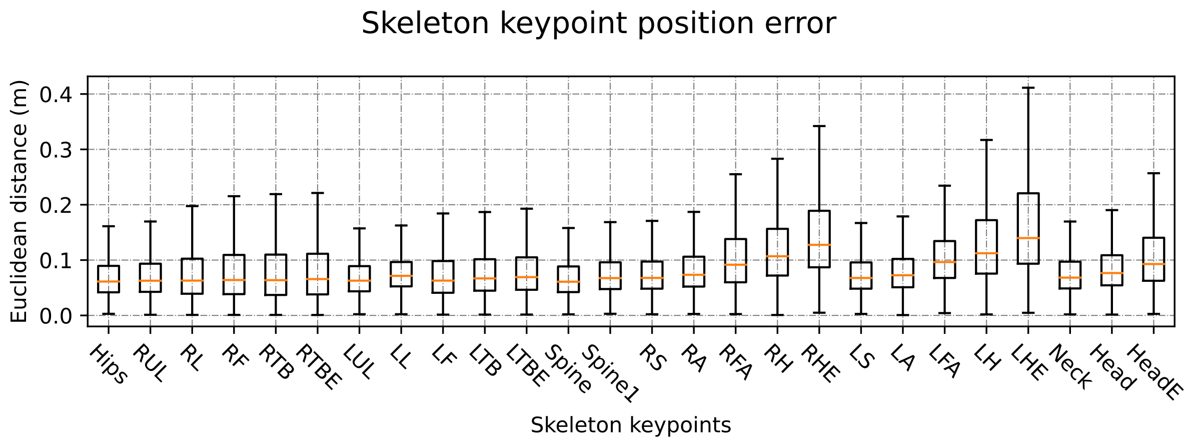


Figure 5.15: Distribution of distances between the target and predicted skeleton keypoints. The predictions were calculated with a PointnetPoseRNN model trained on the entire mmRadPose dataset.

Thresholds	t=0.02	t=0.04	t=0.06	t=0.08	t=0.1	t=0.12	t=0.14	t=0.16	t=0.18	t=0.2
PCK	2.89	15.83	35.51	53.71	67.17	76.54	82.93	87.25	90.22	92.26

Table 5.8: Percent of Correct Keypoints from a PointnetPoseRNN model trained on the entire mmRadPose dataset.

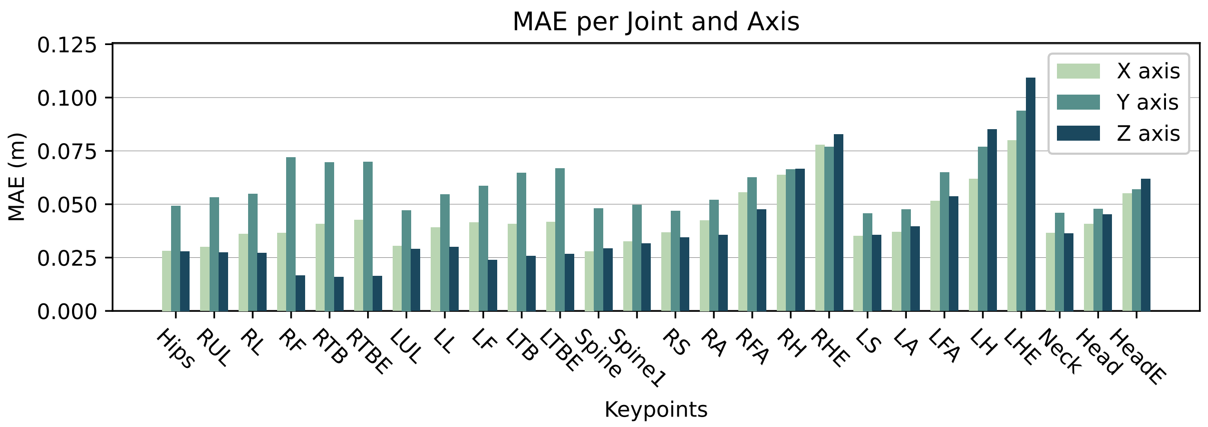


Figure 5.16: Mean Absolute Error (MAE) for each keypoint and axis combination. The Mean Absolute Error was calculated from the predictions of a PointnetPoseRNN model trained on the entire mmRadPose dataset.

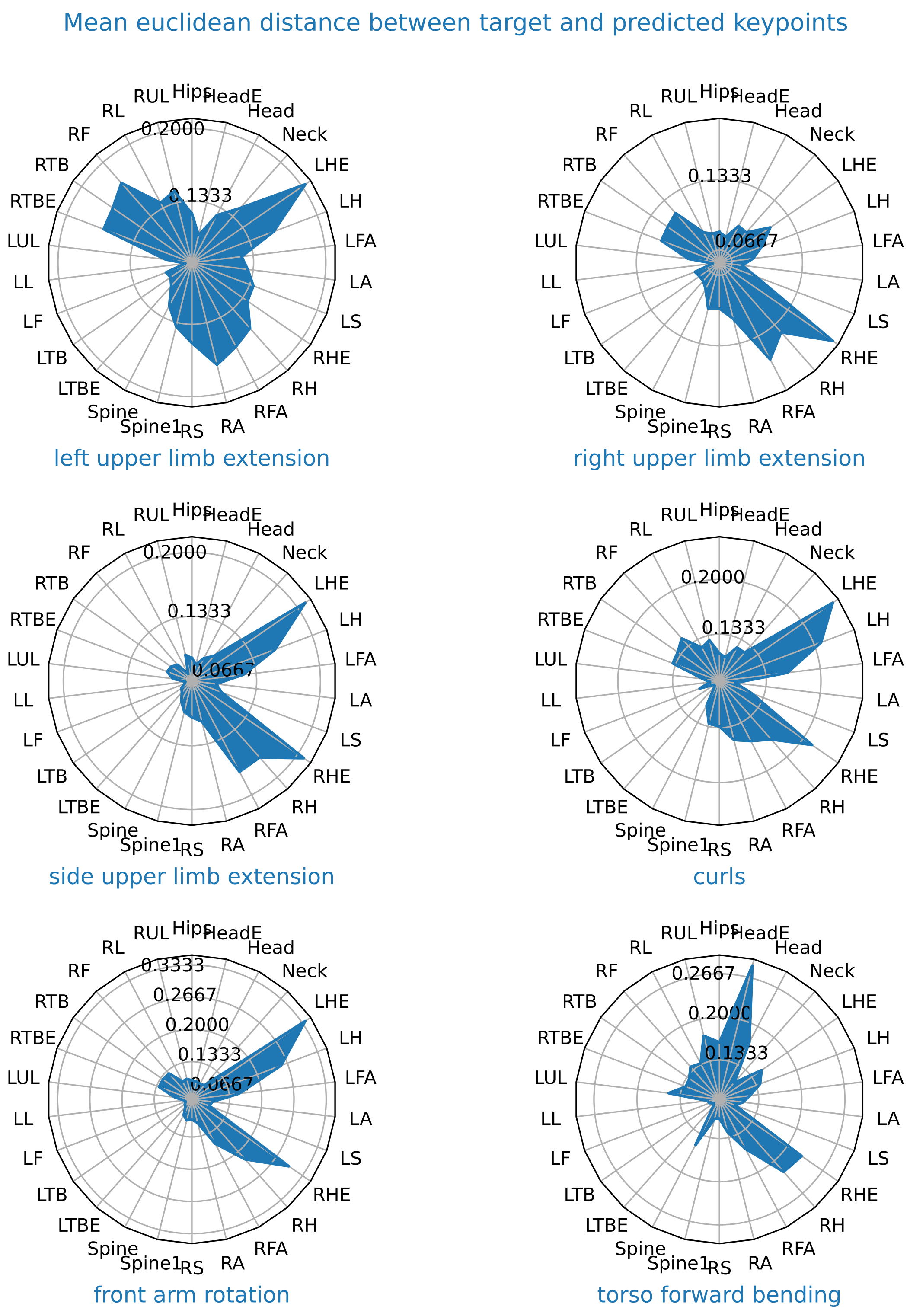


Figure 5.17: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action from the action group 0. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the entire the mmRadPose dataset. Part 1.

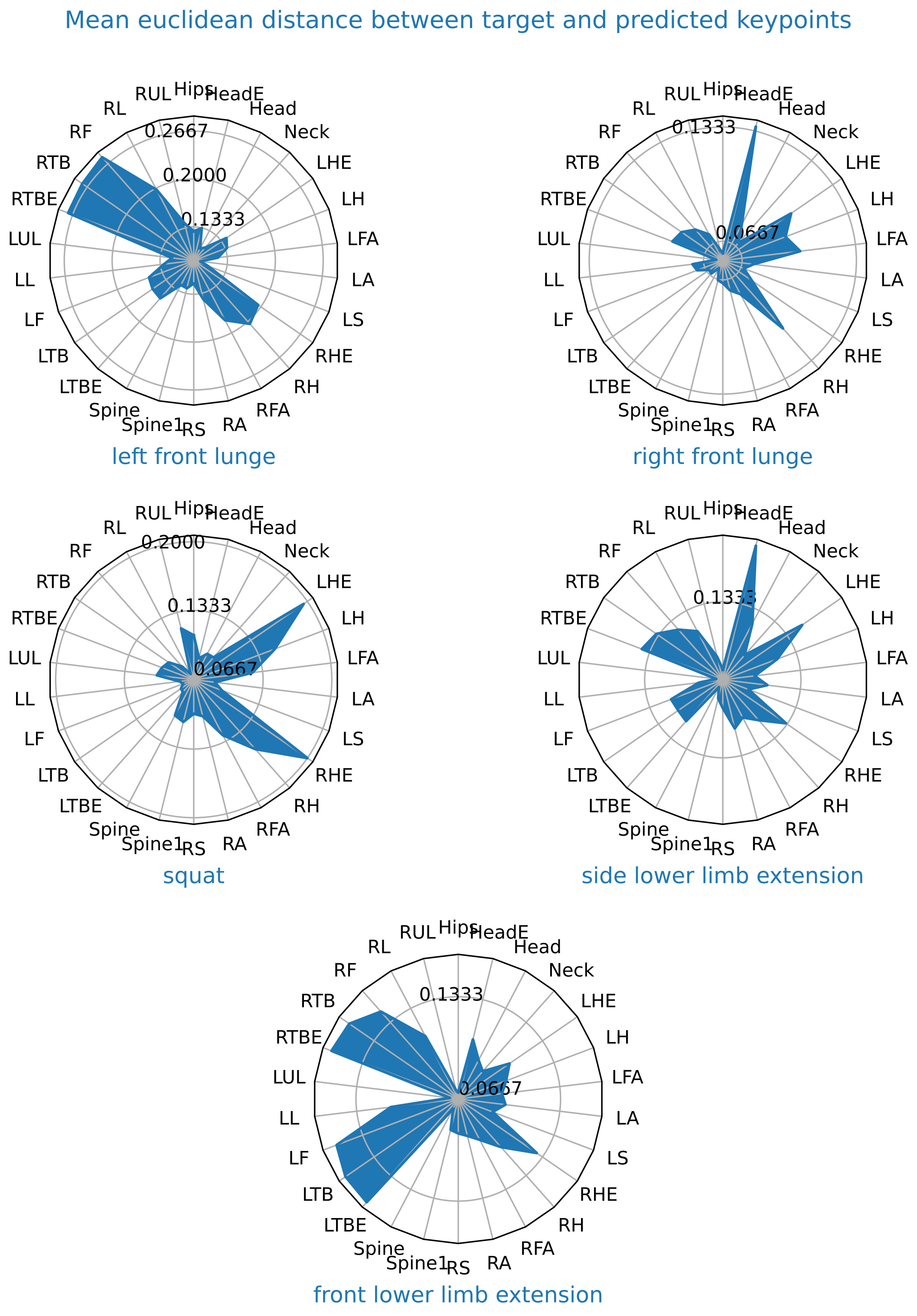


Figure 5.18: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action from the action group 0. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the entire the mmRadPose dataset. Part 2.

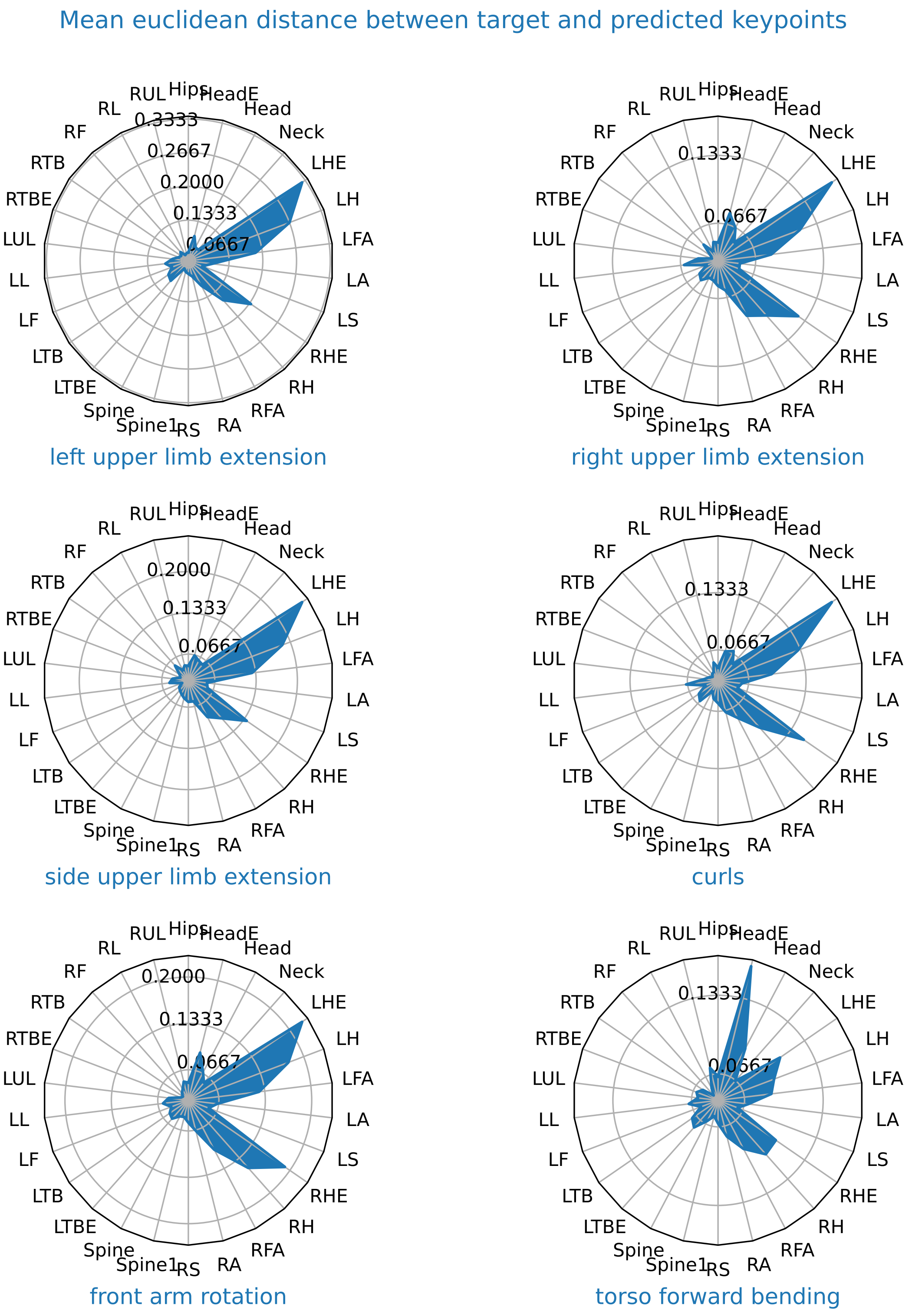


Figure 5.19: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action from the action group 1. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the entire the mmRadPose dataset. Part 1.

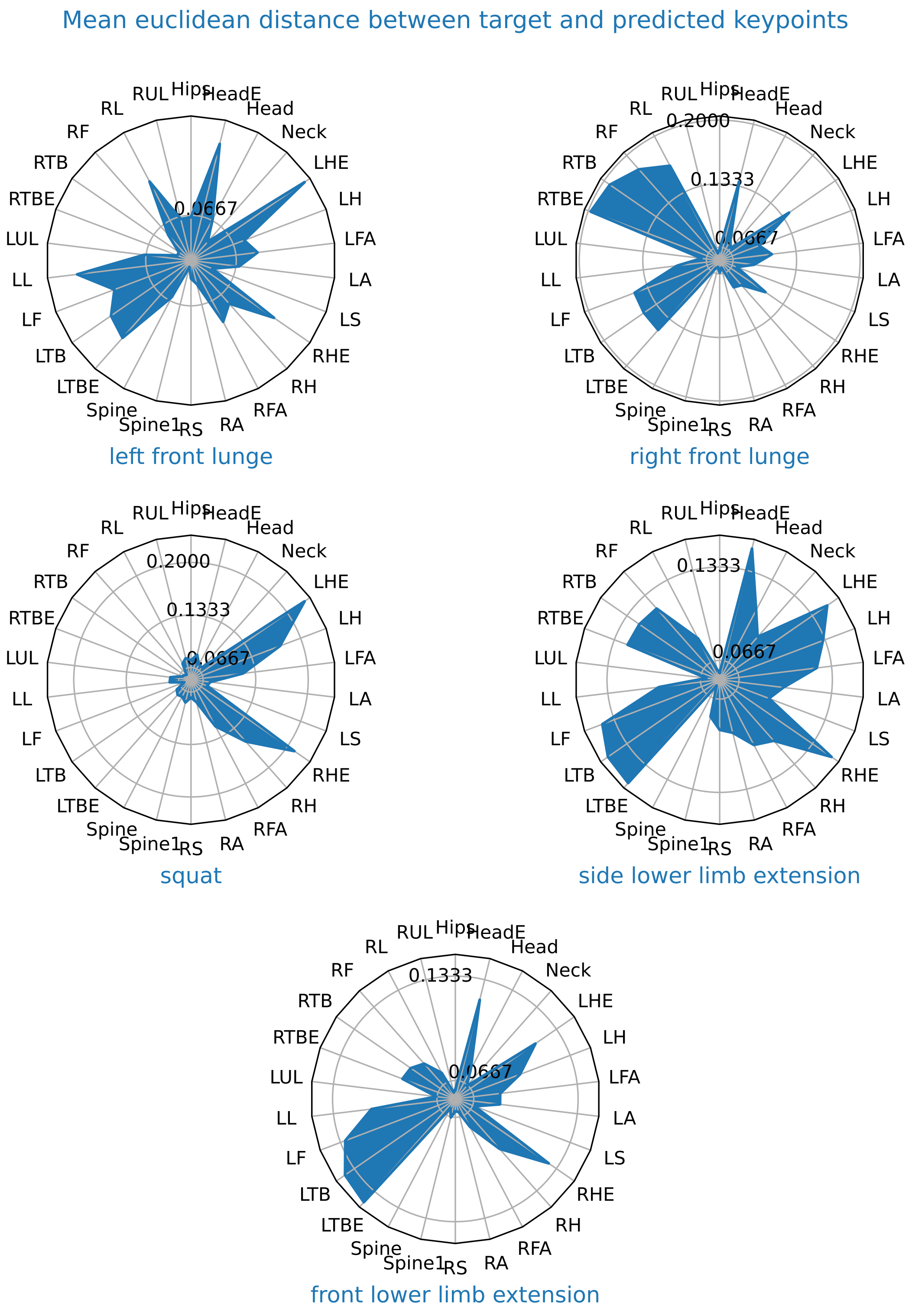


Figure 5.20: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action from the action group 1. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the entire the mmRadPose dataset. Part 2.

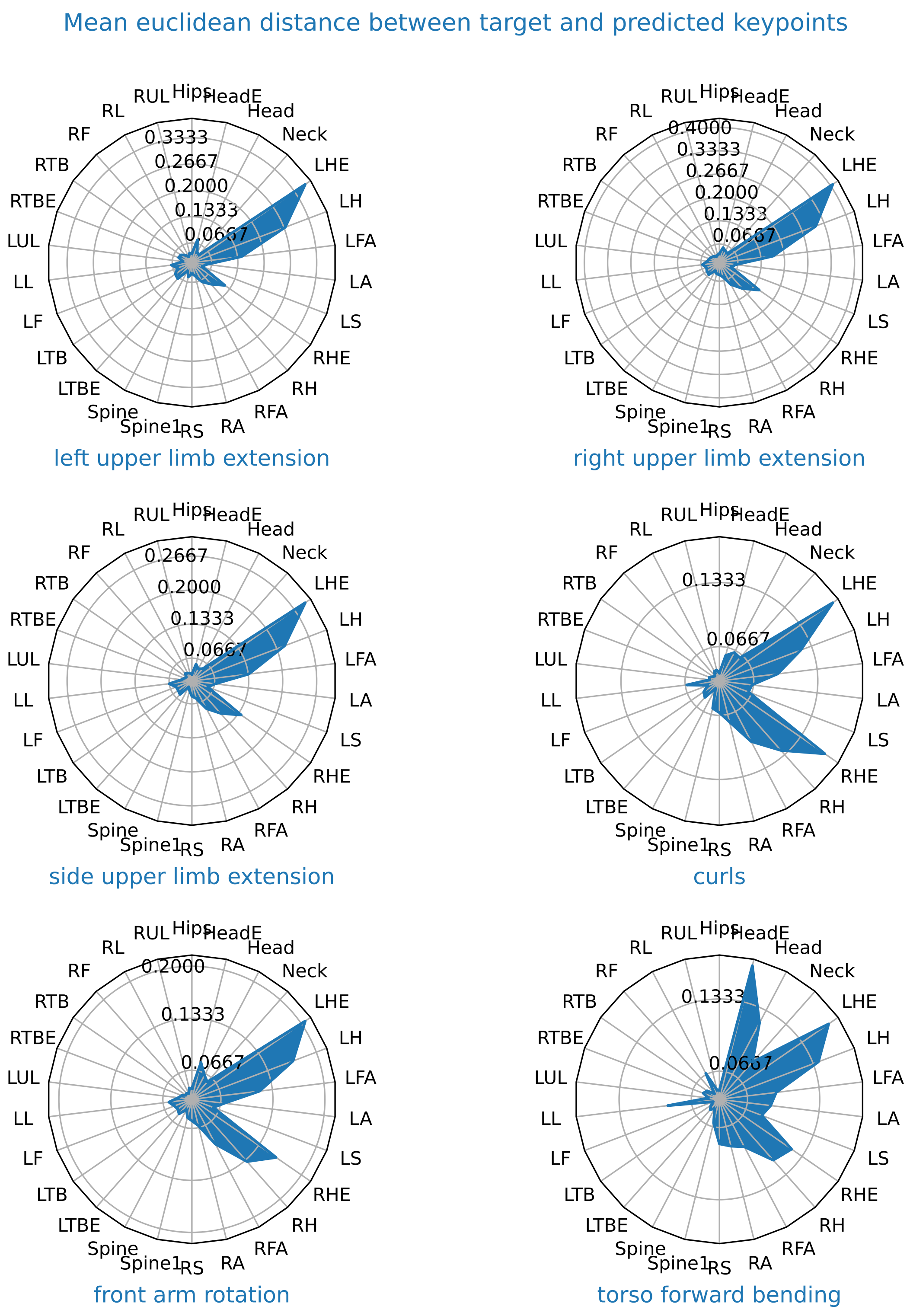


Figure 5.21: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action from the action group 2. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the entire the mmRadPose dataset. Part 1.

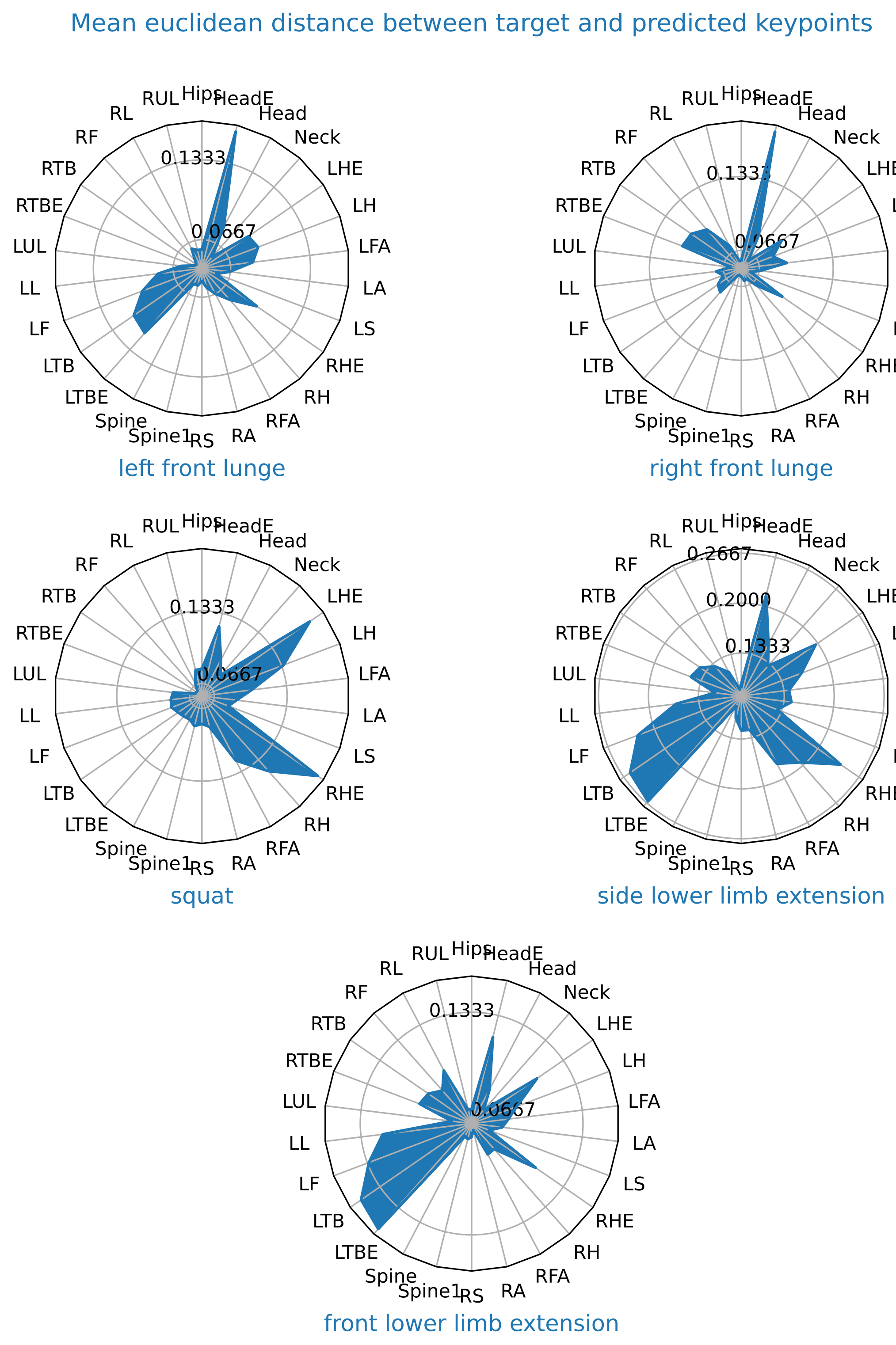


Figure 5.22: Each radar plot depicts the mean euclidean distance between predicted and target keypoints for a specific action from the action group 2. The predicted keypoints were obtained via a PointnetPoseRNN model trained on the entire the mmRadPose dataset. Part 2.

Chapter 6

Discussion

The previous chapter introduced PointnetPose and PointnetPoseRNN performance. In this chapter we discuss the results achieved in the different tasks.

6.1 PointnetPose validation

PointnetPose was validated by training it with the Mars [An21] dataset and comparing it with the results of the model from the same article. PointnetPose performed better, not only according to the PCK curve, but also presented a smaller MPJPE.

However, the results of PointnetPose and PointnetPoseRNN on the mmRadPose dataset, were not nearly as good. PointnetPose went from a MPJPE of 7.26 cm to 14.18 cm. This discrepancy is most likely related to the way the data was split in each dataset. In the Mars dataset, the data was split time-wise. Each recording was divided into training, validation, and testing sets at a ratio of $60\,\%$, $20\,\%$, $20\,\%$ respectively. Therefore, the validation and test data were closely correlated to the training data since the movements were performed by the same persons. On the other hand, the mmRadPose dataset designated the data from two participants exclusively for testing, introducing entirely new data instances for evaluation, which may have contributed to the observed differences in performance. But it encourages overall learning and generalization.

6.2 Preliminary Experiments

PointnetPose and PointnetPoseRNN models were trained using the data from the mmRadPose dataset with the recordings from the action group 0.

The experiments showed that using all features obtained from the radar pointclouds resulted in a better performance. Although there is a correlation between the noise and the signal-to-noise ratio. These features enhanced the quality of the predictions.

It was found that because the T-pose sequences were recorded while the participant sustained a static posture, the pointclouds on the T-pose recordings contained few or no points. This resulted in the network learning to predict a T-pose skeleton when fewer points were present in the input pointcloud. Therefore, excluding the T-pose from training had a positive impact in performance.

In the results of the experiments in the Section was observed that with a threshold of 10 cm, less than 50 % of the keypoints were correctly classified. This fact was corroborated by the MPJPE value close to 15 cm for the four algorithms.

The PointnetPoseRNN models included GRU units to learn from sequences of frames, since radar pointcloud frames are sparse and often there are only a few points per frame. Pointnet-PoseRNN was tested with three different sequence lengths. The results showed that the model with greater sequence length performed the best. However, as described in Section 3.4, the sequence of frames obtained from the mmRadPose dataset are obtained from the past with respect to the current skeleton frame. That means that although using a greater sequence length is beneficial for the amount of data the model receives, in real-time applications, the radar would need to record a number of frames equal or greater than the sequence length before the model can function properly. The PointnetPoseRNN model trained with sequence length of 2 showed no improvement whatsoever in comparison with the PointnetPose model.

6.3 Evaluation of PointnetPoseRNN on mmRadPose

One of the most used metrics for evaluating the accuracy of HPE models is the MPJPE 4.2. This metric measures the mean of the distances between the predicted keypoints and the target keypoints. However, this metric is not enough to determine which of the joints present the greatest deviation.

In order to understand better the functioning of the models. Three PointnetPoseRNN models were trained on a different action group each. And a fourth model was trained on the whole dataset.

The mean of the euclidean distances between each predicted keypoint and the target were shown in radar plots.

6.3.1 Evaluation of PointnetPoseRNN on the action group 0

According to the Figure 5.3 the median of the distances close to the MPJPE of 11.96 cm. However, the hands (RightHand, LeftHand) and the fingers (RightHandEnd, LeftHandEnd) present skewed distributions and therefore the highest error.

Looking at the MAE on the Table 5.4 it shows the highest deviation in the Y axis.

The Figure 5.5 and 5.6 represent the mean distances between the predicted keypoint and target keypoint per action group and action.

The first movement, left upper limb extension, displays the highest error corresponding to the left hand. Similarly, the second movement, right upper limb extension has a high error associated with the right hand. Movements such as side upper limb extension, curls, front arm rotation and squat present a high error in both hands. While movements like side lower limb extension and front lower limb extension present high error in the legs.

6.3.2 Evaluation of PointnetPoseRNN on the action group 1

A PointnetPoseRNN model was trained on the data from the action group 1. The distribution of distances is similar to that in the action group 0. However the PCK is significantly higher for the model trained on action group 2. It is 74.38 % for a threshold of 10 cm in contrast with the PCK of the model trained on action group 1 for the same threshold.

This action group show less distinction between the errors presented for actions such as left upper limb extension and right upper limb extension. Instead it seems to have a high error for the left hand. The participant's orientation with respect to the radar is shown on Figure 3.2. It shows that in the action group 1, the left hand is partially occluded. Which cloud difficult the estimation of the hand location.

Additionally, side lower limb extension and front lower limb extension presented the high errors related to the left leg in comparison with the error present on the right leg.

6.3.3 Evaluation of PointnetPoseRNN on the action group 2

Similarly to the action group 1, the action group 2 presents a high error for the predictions of the left hand. In this almost three times the error of the right hand while performing right upper limb extension. The Figure 6.1 makes a comparison between the predictions of the models on different action groups. The evident increment of the left-hand error suggest that the model is not able to handle the occlusions.

6.3.4 Evaluation of PointnetPoseRNN on all the action groups (0, 1, 2)

The PointnetPoseRNN model trained on the entire mmRadPose dataset, was trained only for only 320 epochs, however it was able to perform similarly as the models trained with independent action

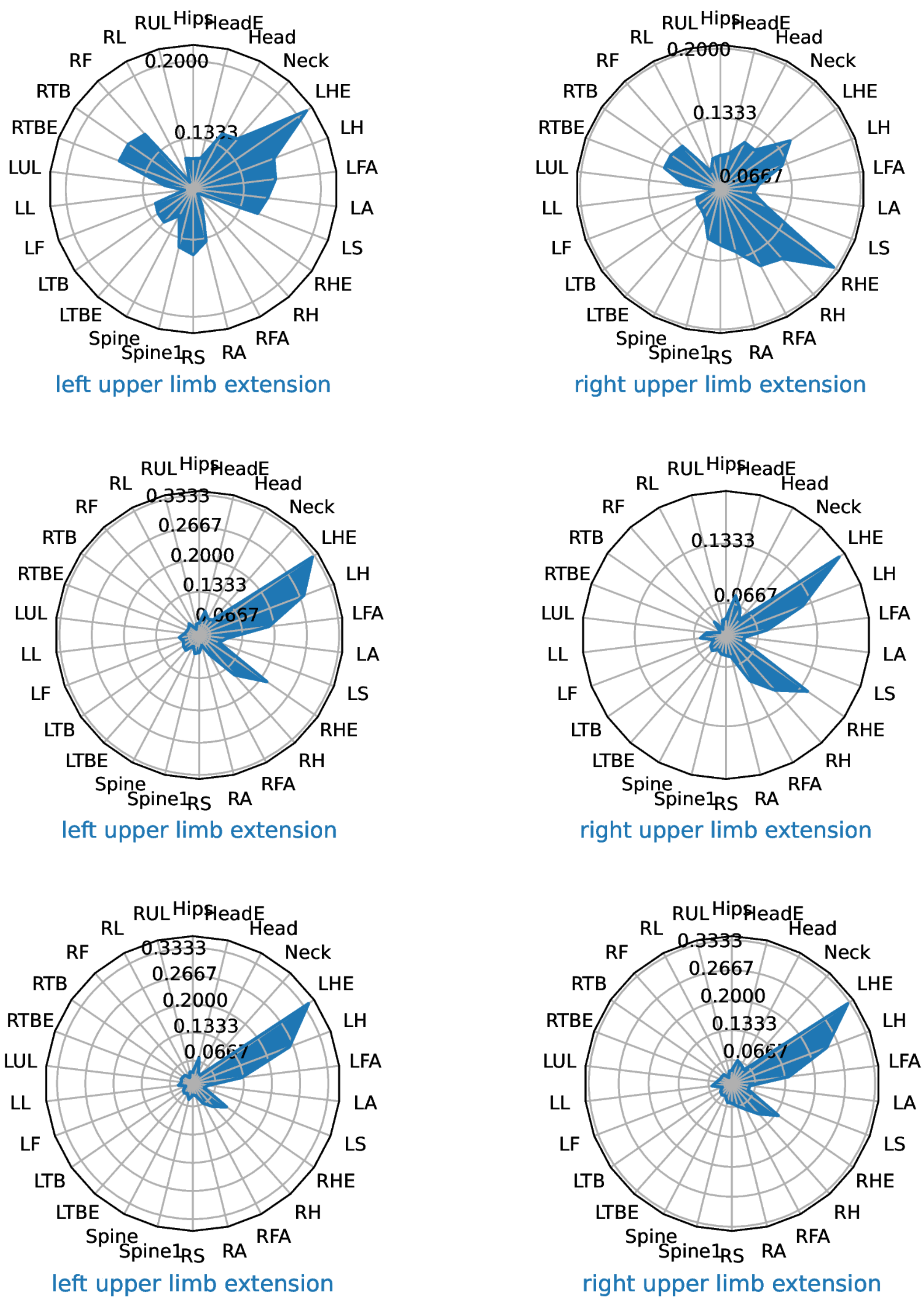


Figure 6.1: Compares the euclidean distances radar plots for the left and right upper limb extension from the action group 0, 1 and 2. The first, second and last row corresponds to the predictions of the PointnetPoseRNN models trained on the action group 1, 2 and 3 respectively.

groups. The capacity of the model proves to be significant.

The PointnetPoseRNN model trained with the whole mmRadPose dataset was able to react to the limbs movements in the same way as the other models did. This conclusion was made taking into account the similarity between the radar plots depicting the distance errors in the predictions and the same plots generated for other model.

Chapter 7

Summary

The aim of this thesis was to evaluate Pointnet based models on performing HPE tasks using mmWave radar pointclouds. Therefore, a study involving twelve participants was conducted to obtain radar pointclouds and pose data. A $60\,\mathrm{GHz}$ FMCW MIMO mmWave radar was used to obtain radar pointclouds from the participants. The participants wore a motion capture suit with reflector markers attached in conjunction with an Optitrack OMC system to obtain 26 body keypoints representing a human skeleton. The data was condensed in the mmRadPose dataset, containing three action groups (0, 1 and 2) where the participants were recorded: facing the radar, turned $45°$ counterclockwise and $90°$ counterclockwise respectively. Each action group contained recordings of the participant performing eleven rehabilitation movements.

PointnetPose, a model derived from the Pointnet [Qi17a] architecture, was developed and validated on data from the Mars [An21] dataset. Another model called PointnetPoseRNN was designed using PointnetPose as a feature extractor followed by an RNN. PointnetPoseRNN accepts a sequence of radar pointclouds as input. Both models were trained for 500 epochs on the mmRadPose dataset on the action group 0 with MSE as loss function. They were tested on the test set composed by two unseen participants during training and PointnetPose achieved am MPJPE of $13.62\,\mathrm{cm}$ while PointnetPoseRNN with a sequence length of 4 achieved an MPJPE of $12.47\,\mathrm{cm}$ on a test set composed by two unseen participants during training.

Three PointnetPoseRNN models with sequence length 10 were trained independently on the three action groups. They reported an MPJPE of $11.96\,\mathrm{cm}$, $8.82\,\mathrm{cm}$ and $8.83\,\mathrm{cm}$ on the action groups 0, 1 and 2 respectively. Lastly another PointnetPoseRNN model was trained using the three action groups simultaneously, achieving an MPJPE of $9.94\,\mathrm{cm}$. During the research was made evident that our models were not able to accurately deal with occlusions using radar data.

In this thesis a dataset for HPE comprising twelve participants was constructed. Additionally, a study protocol for radar and OMC data recording was designed allowing to extend the dataset in the future. In this thesis, the PointnetPose and PointnetPoseRNN architectures were proposed and tested on real data proving the efficiency of using a Pointnet based architecture to perform HPE on mmWave radar generated pointclouds.

Chapter 8

Outlook

The data recorded during the motion study included rehabilitation movements, functional movements, walking, and free-form movements. However, only the rehabilitation movements recorded from three different angles were post-processed and included in the mmRadPose dataset. The performance of PointnetPose and PointnetPoseRNN should be evaluated on the remaining data, with particular focus on movements where the participants transitioned to different locations. Future work should include extending the dataset, including additional participants which increase the generalization ability of the models. Increasing the sample frequency of the Optitrack could lead to less mislabeled markers reducing the amount of effort during the post-processing stage. Utilizing a different marker set with more markers such as Biomech (57) [Opta] could improve the performance especially under occlusions.

For the experiments conducted as part of this thesis, a fixed set of hyperparameters was used. To improve the generalizability of the results, a hyperparameter tuning study should be performed. Although the metrics used in this thesis provided an accurate assessment of the models' performance, they were only able to measure the difference between the skeleton predictions and the target. The design of a metric capable of including keypoints' trajectories in time could be useful to assess the smoothness of the movements of the predicted skeleton.

The original Pointnet [Qi17a] paper demonstrated the robustness of the model against small perturbations like jittering in the pointcloud points. This was not tested during this study. Furthermore, exploring the potential of Pointnet++ [Qi17b] could yield valuable insights into improving pose estimation accuracy.

This research has yet to cover multiple person pose estimation. Only performing single person pose estimation is a great limitation in most scenarios. The possibility of using multiple sensors should also be evaluated.

List of Figures

List of Tables

Bibliography

[An21] Sizhe An and Umit Ogras. “MARS: mmWave-based Assistive Rehabilitation System for Smart Healthcare”. In: *ACM Transactions on Embedded Computing Systems* 20 (Oct. 2021), pp. 1–22. DOI: 10.1145/3477003.

[An22a] Sizhe An, Yin Li, and Umit Ogras. *mRI: Multi-modal 3D Human Pose Estimation Dataset using mmWave, RGB-D, and Inertial Sensors*. 2022. arXiv: 2210.08394 [cs.CV].

[An22b] Sizhe An and Umit Y. Ogras. “Fast and scalable human pose estimation using mmWave point cloud”. In: *Proceedings of the 59th ACM/IEEE Design Automation Conference*. DAC ’22. ACM, July 2022. DOI: 10.1145/3489517.3530522. URL: http://dx.doi.org/10.1145/3489517.3530522.

[Ann16] DataCanary Anna Montoya. *House Prices - Advanced Regression Techniques*. 2016. URL: https://kaggle.com/competitions/house-prices-advanced-regression-techniques.

[Ben21] Miniar Ben Gamra and Moulay A. Akhloufi. “A review of deep learning techniques for 2D and 3D human pose estimation”. In: *Image and Vision Computing* 114 (2021), p. 104282. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2021.104282. URL: https://www.sciencedirect.com/science/article/pii/S0262885621001876.

[Bot98] Léon Bottou. “Online Algorithms and Stochastic Approximations”. In: *Online Learning and Neural Networks*. Ed. by David Saad. revised, oct 2012. Cambridge, UK: Cambridge University Press, 1998. URL: http://leon.bottou.org/papers/bottou-98x.

[Bri67] E. O. Brigham and R. E. Morrow. “The fast Fourier transform”. In: *IEEE Spectrum* 4.12 (1967), pp. 63–70. DOI: 10.1109/MSPEC.1967.5217220.

[Cao19] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2019. arXiv: 1812.08008 [cs.CV].

[Che18] Qiuhui Chen, Chongyang Zhang, Weiwei Liu, and Dan Wang. "SHPD: Surveillance Human Pose Dataset and Performance Evaluation for Coarse-Grained Pose Estimation". In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. 2018, pp. 4088–4092. DOI: 10.1109/ICIP.2018.8451116.

[Che23] Hongren Cheng, Jing Wang, Anran Zhao, Yaping Zhong, Jingli Li, and Liangshan Dong. "Joint graph convolution networks and transformer for human pose estimation in sports technique analysis". In: *Journal of King Saud University - Computer and Information Sciences* 35.10 (2023), p. 101819. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2023.101819. URL: https://www.sciencedirect.com/science/article/pii/S1319157823003737.

[Cho14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

[Chu14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. arXiv: 1412.3555 [cs.NE].

[Cle16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. *Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)*. 2016. arXiv: 1511.07289 [cs.LG].

[Cuk12] Will Cukierski. *Titanic - Machine Learning from Disaster*. 2012. URL: https://kaggle.com/competitions/titanic.

[Cyb89] George V. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals and Systems* 2 (1989), pp. 303–314. URL: https://api.semanticscholar.org/CorpusID:3958369.

[Den24] Yicheng Deng, Cheng Sun, Yongqi Sun, and Jiahui Zhu. "3D human pose estimation based on 2D–3D consistency with synchronized adversarial training". In: *Robotics and Autonomous Systems* 175 (2024), p. 104677. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2024.104677. URL: https://www.sciencedirect.com/science/article/pii/S0921889024000605.

[Est96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.

[Fu23] Huichen Fu, Junwei Gao, and Huabo Liu. "Human pose estimation and action recognition for fitness movements". In: *Computers & Graphics* 116 (2023), pp. 418–426. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2023.09.008. URL: https://www.sciencedirect.com/science/article/pii/S0097849323002315.

[Fuk69] Kunihiko Fukushima. "Visual Feature Extraction by a Multilayered Network of Analog Threshold Elements". In: *IEEE Transactions on Systems Science and Cybernetics* 5.4 (1969), pp. 322–333. DOI: 10.1109/TSSC.1969.300225.

[Gei16] Florian Geiselhart, Michael Otto, and Enrico Rukzio. "On the Use of Multi-Depth-Camera Based Motion Tracking Systems in Production Planning Environments". In: *Procedia CIRP* 41 (2016). Research and Innovation in Manufacturing: Key Enabling Technologies for the Factories of the Future - Proceedings of the 48th CIRP Conference on Manufacturing Systems, pp. 759–764. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2015.12.088. URL: https://www.sciencedirect.com/science/article/pii/S2212827115011671.

[Goo16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Has24] Kareeb Hasan, Beng Oh, Nithurshan Nadarajah, and Mehmet Rasit Yuce. "mm-CasGAN: A cascaded adversarial neural framework for mmWave radar point cloud enhancement". In: *Information Fusion* 108 (2024), p. 102388. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2024.102388. URL: https://www.sciencedirect.com/science/article/pii/S1566253524001660.

[Hau85] John Haugeland. *Artificial intelligence: the very idea*. en. Cambridge, Mass: MIT Press, 1985. ISBN: 978-0-262-08153-5.

[He15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[Hoc97] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[Hor91] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural Networks* 4.2 (1991), pp. 251–257. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(91)90009-T. URL: https://www.sciencedirect.com/science/article/pii/089360809190009T.

[Ins22] Texas Instruments. *IWR6843AOP Single-Chip 60- to 64-GHz mmWave Sensor Antennas-On-Package (AOP)*. TJA1043. 2022. URL: https://www.ti.com/lit/ds/symlink/iwr6843aop.pdf (visited on 04/22/2024).

[Jab15] Haider Khalaf Jabbar and Rafiqul Zaman Khan. “Survey on development of expert system in the areas of Medical, Education, Automobile and Agriculture”. In: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2015, pp. 776–780.

[Kin17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[Kla17] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. *Self-Normalizing Neural Networks*. 2017. arXiv: 1706.02515 [cs.LG].

[Koc22] Mertkan Koca, Gokhan Gurbilek, and Sinem Coleri. “mmWave channel model for intra-vehicular wireless sensor networks”. In: *Ad Hoc Networks* 135 (2022), p. 102932. ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2022.102932. URL: https://www.sciencedirect.com/science/article/pii/S1570870522001160.

[Kur90] Raymond Kurzweil. *The Age of Intelligent Machines*. en. MIT Press, 1990. ISBN: 9780262610797.

[LeC15] Yann LeCun, Y. Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.

[LeC89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4 (1989), pp. 541–551. DOI: 10.1162/neco.1989.1.4.541.

[Lec89] Yann Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L.D. Jackel. “Backpropagation applied to handwritten zip code recognition”. English (US). In: *Neural Computation* 1.4 (1989), pp. 541–551. ISSN: 0899-7667.

[LeC98] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus -Robert Müller. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade*. Ed. by Genevieve B. Orr and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 9–50. ISBN: 978-3-540-49430-0. DOI: 10.1007/3-540-49430-8_2. URL: https://doi.org/10.1007/3-540-49430-8_2.

[Lee22] Shih-Po Lee, Niraj Prakash Kini, Wen-Hsiao Peng, Ching-Wen Ma, and Jenq-Neng Hwang. *HuPR: A Benchmark for Human Pose Estimation Using Millimeter Wave Radar*. 2022. arXiv: 2210.12564 [cs.CV].

[Liu23] Tingyu Liu, Chenyi Weng, Lei Jiao, Jun Huang, Xiaoyu Wang, Zhonghua Ni, and Baicun Wang. "Toward fast 3D human activity recognition: A refined feature based on minimum joint freedom model (Mint)". In: *Journal of Manufacturing Systems* 66 (2023), pp. 127–141. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2022.11.009. URL: https://www.sciencedirect.com/science/article/pii/S027861252200200X.

[Luo22] Yanmin Luo, Zhilong Ou, Tianjun Wan, and Jing-Ming Guo. "FastNet: Fast high-resolution network for human pose estimation". In: *Image and Vision Computing* 119 (2022), p. 104390. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2022.104390. URL: https://www.sciencedirect.com/science/article/pii/S0262885622000191.

[Lyu24] Lu Lyu and Yong Huang. "Sports activity (SA) recognition based on error correcting output codes (ECOC) and convolutional neural network (CNN)". In: *Heliyon* 10.6 (2024), e28258. ISSN: 2405-8440. DOI: https://doi.org/10.1016/j.heliyon.2024.e28258. URL: https://www.sciencedirect.com/science/article/pii/S2405844024042890.

[Maa13] Andrew L. Maas. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: 2013. URL: https://api.semanticscholar.org/CorpusID:16489696.

[Mar22] Enrico Martini, Michele Boldo, Stefano Aldegheri, Nicola Valè, Mirko Filippetti, Nicola Smania, Matteo Bertucco, Alessandro Picelli, and Nicola Bombieri. "Enabling Gait Analysis in the Telemedicine Practice through Portable and Accurate 3D Human Pose Estimation". In: *Computer Methods and Programs in Biomedicine* 225 (2022), p. 107016. ISSN: 0169-2607. DOI: https://doi.org/10.1016/j.cmpb.2022.107016. URL: https://www.sciencedirect.com/science/article/pii/S0169260722003984.

[Mer19] Marco Mercuri, Ilde Rosa Lorato, Yao-Hong Liu, Fokko Wieringa, Chris Van Hoof, and Tom Torfs. "Vital-sign monitoring and spatial tracking of multiple people using a contactless radar-based sensor". In: *Nature Electronics* 2.6 (2019), pp. 252–262.

[Mni13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

[Mow22] Md Munjure Mowla, Iftekhar Ahmad, Daryoush Habibi, Quoc Viet Phung, and M. Ishtiaque Aziz Zahed. "Green traffic backhauling in next generation wireless communication networks incorporating FSO/mmWave technologies". In: *Computer Communications* 182 (2022), pp. 223–237. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2021.11.006. URL: https://www.sciencedirect.com/science/article/pii/S0140366421004333.

[Opta] Optitrack. *Biomech (57) | v3.1 | EXTERNAL OptiTrack Documentation*. Accessed on April 28, 2024. URL: https://docs.optitrack.com/movement-sciences/movement-sciences-markersets/biomech-57.

[Optb] Optitrack. *Conventional (39) | v3.1 | EXTERNAL OptiTrack Documentation*. Accessed on April 4, 2024. URL: https://docs.optitrack.com/markersets/full-body/conventional-39.

[Qi17a] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV].

[Qi17b] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/d8bf84be3800d12f74d8b05e9b89836f-Paper.pdf.

[Ric14] M. A. Richards. *Fundamentals of radar signal processing*. en. Second edition. New York: McGraw-Hill Education, 2014. ISBN: 978-0-07-179833-4.

[Ros57] F. Rosenblatt. *The perceptron - A perceiving and recognizing automaton*. Tech. rep. 85-460-1. Cornell Aeronautical Laboratory, Jan. 1957.

[Ros58] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65 6 (1958), pp. 386–408. URL: https://api.semanticscholar.org/CorpusID:12781225.

[Rum86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323 (1986), pp. 533–536. URL: https://api.semanticscholar.org/CorpusID:205001834.

[Rus14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Li Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115 (Sept. 2014). DOI: 10.1007/s11263-015-0816-y.

[Rus16] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. en. Third edition, Global edition. Prentice Hall series in artificial intelligence. Boston Columbus Indianapolis New York San Francisco Upper Saddle River Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo: Pearson, 2016. ISBN: 978-0-13-604259-4 978-1-292-15396-4.

[Sch19] Nicolas Scheiner, Nils Appenrodt, Jürgen Dickmann, and Bernhard Sick. "A Multi-Stage Clustering Framework for Automotive Radar Data". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 2060–2067. DOI: 10.1109/ITSC.2019.8916873.

[Sea80] John R. Searle. "Minds, brains, and programs". In: *Behavioral and Brain Sciences* 3.3 (1980), pp. 417–424. DOI: 10.1017/S0140525X00005756.

[Sen20] Arindam Sengupta, Feng Jin, Renyuan Zhang, and Siyang Cao. "mm-Pose: Real-Time Human Skeletal Posture Estimation Using mmWave Radars and CNNs". In: *IEEE Sensors Journal* 20.17 (Sept. 2020), pp. 10032–10044. DOI: 10.1109/jsen.2020.2991741. URL: https://doi.org/10.1109%2Fjsen.2020.2991741.

[Shi20] Weijing Shi and Ragunathan Rajkumar. "Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud". In: *CoRR* abs/2003.01251 (2020). arXiv: 2003.01251. URL: https://arxiv.org/abs/2003.01251.

[Sim15] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[Sin23] Rahul Singh, Avinash Sharma, Neha Sharma, and Rupesh Gupta. "Impact of Adam, Adadelta, SGD on CNN for White Blood Cell Classification". In: *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2023, pp. 1702–1709. DOI: 10.1109/ICSSIT55814.2023.10061068.

[Tos14] Alexander Toshev and Christian Szegedy. “DeepPose: Human Pose Estimation via Deep Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[Tou23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL].

[TUR50] A. M. TURING. “I.—COMPUTING MACHINERY AND INTELLIGENCE”. In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. URL: https://doi.org/10.1093/mind/LIX.236.433.

[Uhl22] Scott Uhlrich, Antoine Falisse, Łukasz Kidziński, Julie Muccini, Michael Ko, Akshay Chaudhari, Jennifer Hicks, and Scott Delp. *OpenCap: 3D human movement dynamics from smartphone videos*. July 2022. DOI: 10.1101/2022.07.07.499061.

[Wol] Christian Wolff. *Intrapulse Modulation and Pulse Compression*. URL: https://www.radartutorial.eu/08.transmitters/Intrapulse%20Modulation.en.html.

[Wu15] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. “3D ShapeNets: A deep representation for volumetric shapes”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920. DOI: 10.1109/CVPR.2015.7298801.

[Xu22] Wei Xu, Donghai Xiang, Guotai Wang, Ruisong Liao, Ming Shao, and Kang Li. “Multiview Video-Based 3-D Pose Estimation of Patients in Computer-Assisted Rehabilitation Environment (CAREN)”. In: *IEEE Transactions on Human-Machine Systems* 52.2 (2022), pp. 196–206. DOI: 10.1109/THMS.2022.3142108.

[Yan23a] Baiju Yan, Peng Wang, Lidong Du, Xianxiang Chen, Zhen Fang, and Yirong Wu. “mmGesture: Semi-supervised gesture recognition system using mmWave radar”. In: *Expert Systems with Applications* 213 (2023), p. 119042. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.119042. URL: https://www.sciencedirect.com/science/article/pii/S0957417422020607.

[Yan23b] Jianfei Yang, He Huang, Yunjiao Zhou, Xinyan Chen, Yuecong Xu, Shenghai Yuan, Han Zou, Chris Xiaoxuan Lu, and Lihua Xie. *MM-Fi: Multi-Modal Non-Intrusive 4D Human Dataset for Versatile Wireless Sensing*. 2023. arXiv: 2305.10345 [eess.SP].

[Zho17] Yin Zhou and Oncel Tuzel. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. 2017. arXiv: 1711.06396 [cs.CV].

[Zho24] Lu Zhou, Yingying Chen, and Jinqiao Wang. "SlowFastFormer for 3D human pose estimation". In: *Computer Vision and Image Understanding* 243 (2024), p. 103992. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2024.103992. URL: https://www.sciencedirect.com/science/article/pii/S1077314224000730.

[Zho88] Zhou and Chellappa. "Computation of optical flow using a neural network". In: *IEEE 1988 International Conference on Neural Networks*. 1988, 71–78 vol.2. DOI: 10.1109/ICNN.1988.23914.

[Zhu24a] Chen Zhu, Zhouxiang Zhao, Zejing Shan, Lijie Yang, Sijie Ji, Zhaohui Yang, and Zhaoyang Zhang. "Robust target detection of intelligent integrated optical camera and mmWave radar system". In: *Digital Signal Processing* 145 (2024), p. 104336. ISSN: 1051-2004. DOI: https://doi.org/10.1016/j.dsp.2023.104336. URL: https://www.sciencedirect.com/science/article/pii/S1051200423004311.

[Zhu24b] Yean Zhu, Meirong Xiao, Yijun Xie, Zepu Xiao, Guoqiang Jin, and Lang Shuai. "In-bed human pose estimation using multi-source information fusion for health monitoring in real-world scenarios". In: *Information Fusion* 105 (2024), p. 102209. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2023.102209. URL: https://www.sciencedirect.com/science/article/pii/S1566253523005250.

.1 Study protocol

Data Recording Protocol

Study Purpose

The purpose of this study is to record biomechanics data from healthy participants that can be used to perform human pose estimation tasks.

Objectives

- Obtain biomechanics and radar data from healthy participants performing rehabilitation movements, walking and functional movements.
- Record information about the participant's joint locations using an Optical Motion Capture (OMC) Optitrack system.
- Record radar datacubes and obtain radar pointclouds from the participant's movements.
- Create a dataset for human pose estimation.

Materials

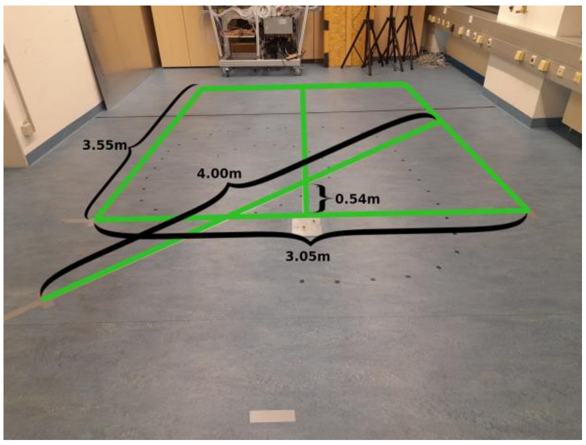
- ❑ Optitrack system (12 cameras)
- ❑ Optitrack calibration utils
- ❑ Optitrack marker suit (+39 markers)
- ❑ Computer equipped with Motive 2.3.1
- ❑ Radar box (mmWave radar + Kinect)
- ❑ 4 styrodur columns + 4 corner reflectors + 4 markers
- ❑ Laptop equipped with the radar triggering software and Natnet 3.1.0
- ❑ Ethernet cable
- ❑ 1 chair

Characteristics of the Recording Room

The recordings will be performed in a designated room equipped with an Optitrack system with 12 cameras distributed around the room. The room's dimensions are 10.60m length, 4.72m width and 3.27m height. An area of 3.55m length by 3.05m width was marked in the middle of the room. This will be the volume of interest covered by the radar and the OMC system.

The participant must perform all the movements within this area. In the area are marked 3 lines signaling 0°, 45° and 90° angles, this will serve as a guide for the participants. The radar was located on a table 2.92m in from the area of interest. The table height is 0.74m and the relative height of the radar with respect to the table is 0.32m.

Original Image of the room



Room with floor mark measurements.

System Setup

A radar box equipped with an mmwave radar, and a Microsoft Kinect sensor will be used to obtain RGB, depth, infra-red and radar data. The radar datacubes will be later processed to obtain pointclouds that will serve as input data for Human Pose Estimation models.

An OMC Optitrack device was installed in the room to obtain the participant's joint locations. The OMC was connected to 12 cameras distributed around the room. The use of this system provides a better accuracy of the estimated joints. It is also more robust to occlusions than single camera systems. The Optitrack system is controlled by the Motive software (version 2.3.1) installed in the computer located in the room.

The NatNet 3.1.0 library will be used to trigger the Optitrack remotely from our main system. NatNet will communicate to a server from Motive to control the Optitrack's recordings. To achieve a real-time synchronization with the radar box along with NatNet, a hardware trigger controlled by the radar box was used. The main system triggers the radar, Kinect and optitrack recordings at a frequency of 15Hz.

Preparations

- ❑ Turn on the computer and the laptop.
- ❑ Connect the radar box to power.
- ❑ Connect the radar's USB serial to the laptop. Connect the hardware trigger pins to the Optihub (Optitrack hub for IO communication). Connect the Optitrack computer to the laptop through an Ethernet cable.
- ❑ Set Optitrack streaming IP in Motive to match the laptop IP.
- ❑ Calibrate Optitrack system and the radar.
- ❑ Make a test recording to test the radar and OMC synchronization.

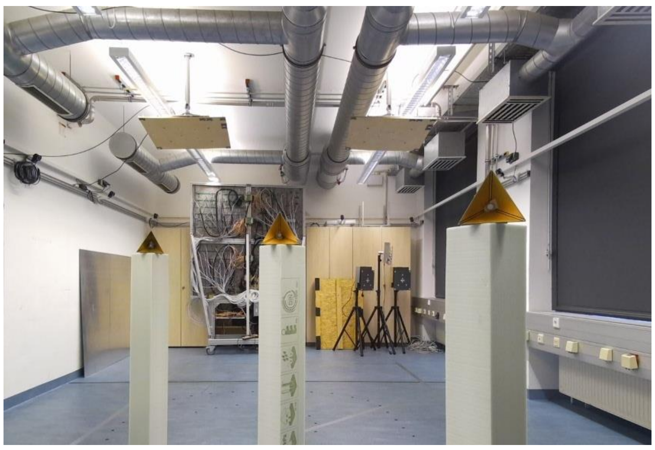
Optitrack Calibration

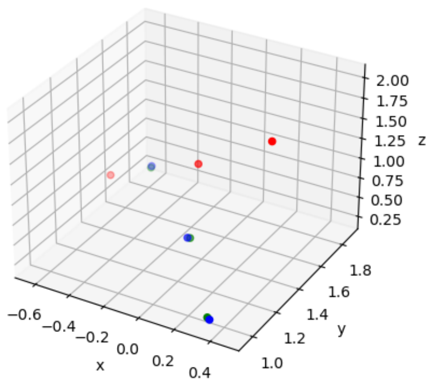
In the beginning the Optitrack sensors are calibrated. The whole process takes at most 15 minutes. The Optitrack is calibrated using a wand with 3 markers and moving it around the room. It will synchronize the marker detections from all cameras and set the space 3D origin.

Refer to https://docs.optitrack.com/motive/calibration for more information about the Optitrack's calibration.

Radar, Kinect and Optitrack Calibration

The purpose of this calibration is to find a transformation matrix that allows to transform the Optitrack's and Kinect's coordinates to those from the radar.


Corner reflectors and Optitrack markers are used for calibration.



The red dots show the detections from Optitrack while the blue dots show the detections from the radar.

Three corner reflectors with Optitrack markers inside, are placed on top of styrodur columns that do not reflect the radar signals. The coordinates of the markers are obtained from the optitrack while the coordinates of the center of the corner reflectors are obtained from the radar. Then a Procrustes Analysis produces the transformation matrix to the Optitrack data. A similar process is done to calibrate the Kinect with the radar.

Participants

This study will be conducted with healthy participants capable of performing the movements for a relatively long period of time. All subjects will be asked to sign a confirmation agreement and were explained about the purpose of the study and how the recording would be done. We also gathered some additional data like the age (years), the height (meters), weight (Kg), etc.

Study Procedure

In the beginning of the study the Optitrack and the radar must be calibrated once. Then we proceed to record data from each participant. Each session lasts from 2 to 3 hours. Therefor each participant will apply for 3 hours recording time slot. Each subject will be required to read and agree with an informed consent document before been part of the study. We will keep record of the name of the participant, age, sex, height, weight and assign a unique identifier for each participant. From the participants will be recorded RGB and depth data, datacubes from the radar and a *.tak* file from Motive containing the participants tracking data.

Recording Session

In the beginning of each recording session the participant must wear the marker's suit. Then all the markers must be checked and relocated if necessary, following the skeleton marker set Conventional 39. This process should take about 15 minutes. The participant will make a T-pose for the Optitrack to fit a new skeleton from the participant's markers detection. The subject will stand at 3m from the radar and perform each of the rehabilitation, walking and free movements describe next.

Movements to study

Each participant will perform a static T-pose and 10 rehabilitation movements in 3 different angles (0°, 45° and 90°) with respect to the radar, walking in a straight line (0°, 45° and 90°), 5 functional movements and 1 free movement. In total 43 recordings will be performed by each participant.

Each recording will last for at least 30 seconds. Before each recording, the instructor will explain the movement to the participant. The participant will start each recording with a T-pose to facilitate the skeleton fitting and start performing the movement after the instructor signal. In total the whole process of explaining the movement, recording and solving errors should take a maximum of 2 minutes per recording.

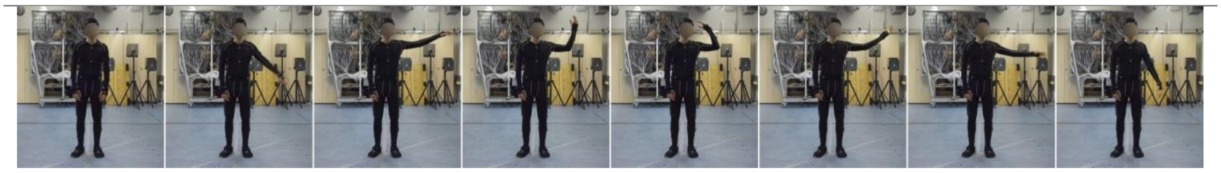
Rehabilitation Movements

For each participant a static T-pose for 10 seconds and 10 rehabilitation movements for 36 seconds will be recorded.

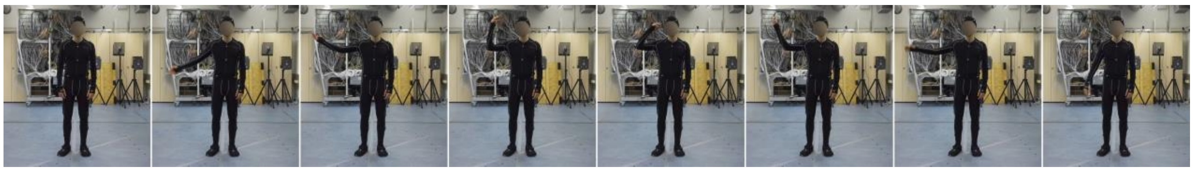
All the movements will be recorded once with the participant facing the radar box (0°). Then the participant will turn 45° to his left and the movements will be recorded again. And finally, the participant will turn 45° and his orientation will be 90° from what it was initially, and the movements will be recorded once more.

Each recording will last for 36 seconds, starting with a T-pose and then performing the indicated movement.

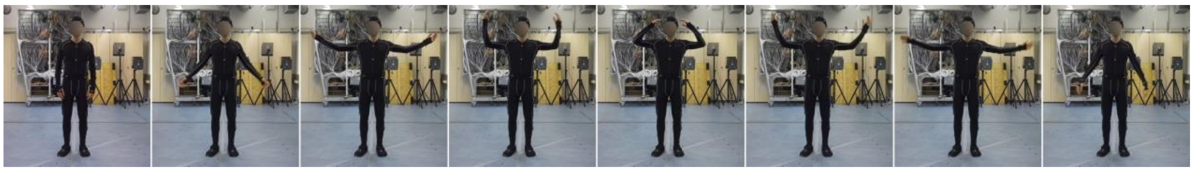
Code	**Action Name**
A1	**Left upper limb extension**


Starting with the left arm close to the body, the participant will raise the arm straight up until it reaches shoulder height and then bend the arm upward.

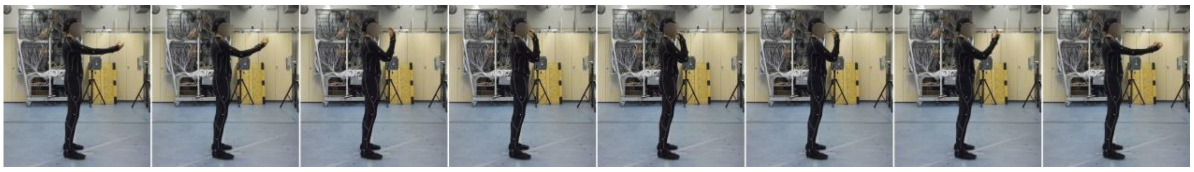
A2 **Right upper limb extension**


Analog to A1 but using the right arm.

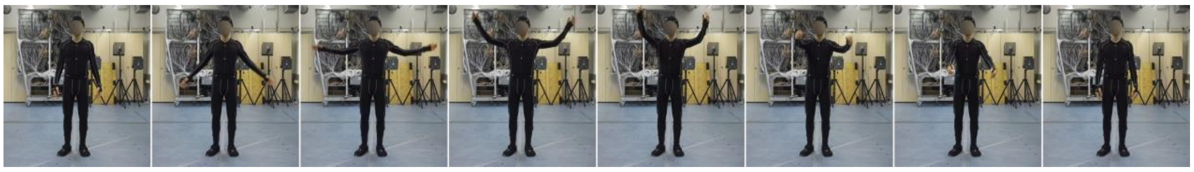
A3 **Side upper limb extension**


Analog to A1 but using both arms at the same time.

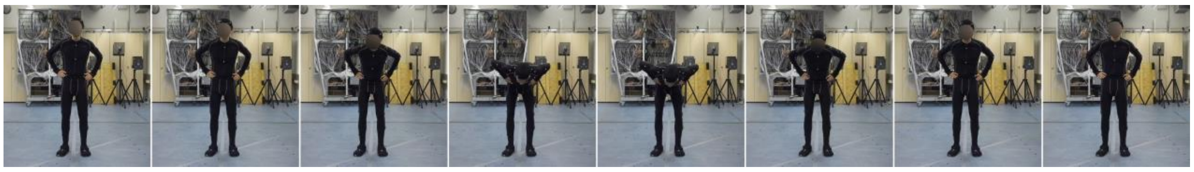
A4 **Curls**


Starting with the arms extended to the front, the participant will bring his hands close to his shoulders without moving the elbows.

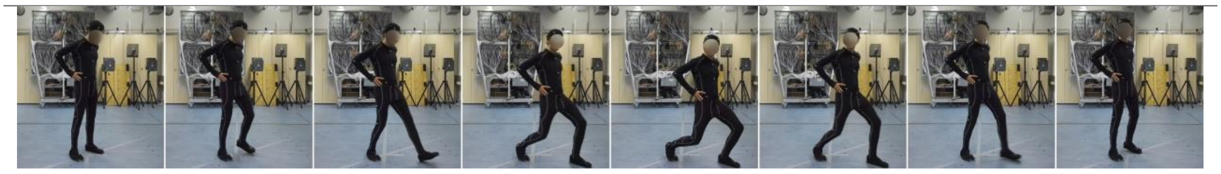
A5 **Front arm rotation**



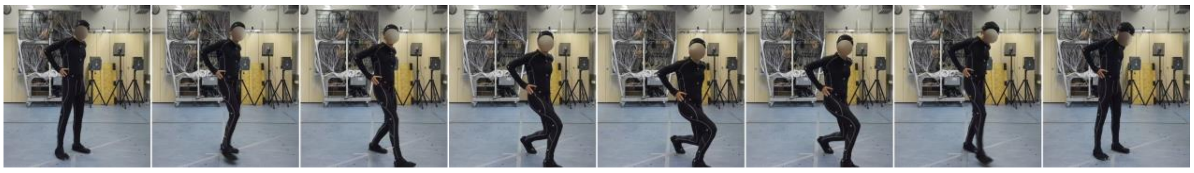
A6 **Torso forward bending**


Starting with the arms on the hips, the participant will bend his torso forward.

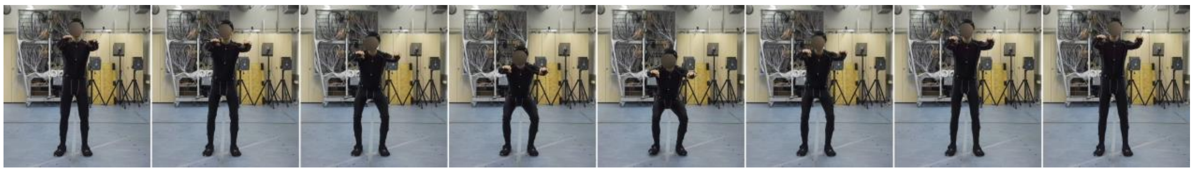
A7 **Left front lunge**



A8 **Right front lunge**

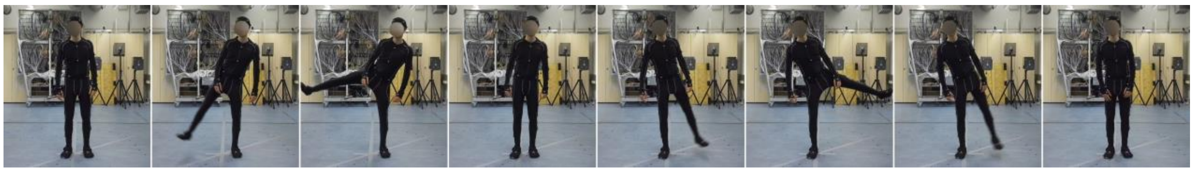


A9 **Squat**



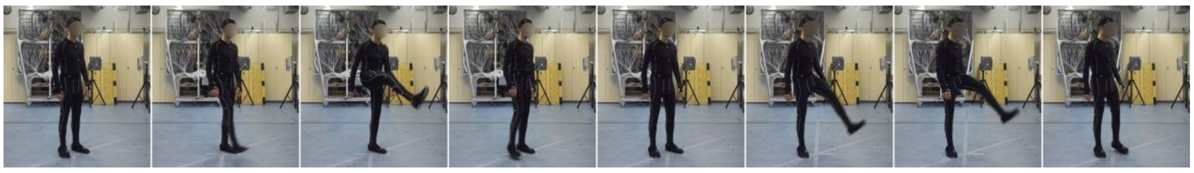
Starting with the arms extended to the front, the participant will perform a squat.

A10 **Side lower limb extension**



With the arms hanging in front of the body, the participant will alternately lift one leg to one side and then the other.

A11 **Front lower limb extension**



With the arms hanging next to the body, the participant will alternately lift one leg to the front and then the other.

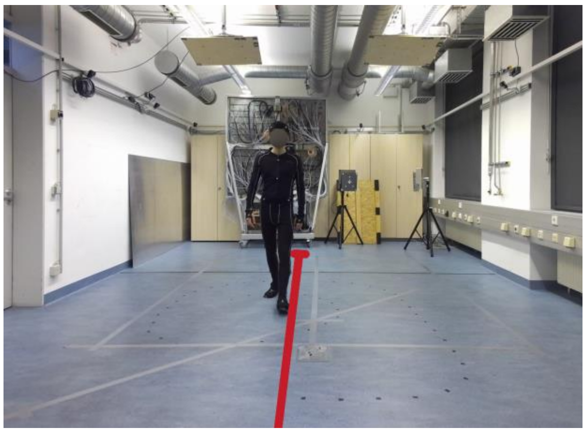
Walking

After the rehabilitation movements the participants will walk back and forth in a straight line each in a different angle (0°, 45°and 90°) with respect to the radar.

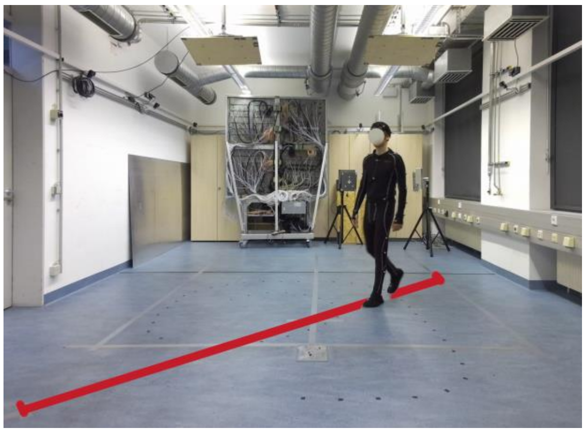
The markings on the floor will indicate the direction and the length of the walk.

Each recording will last for 1 minute starting with a T-pose.

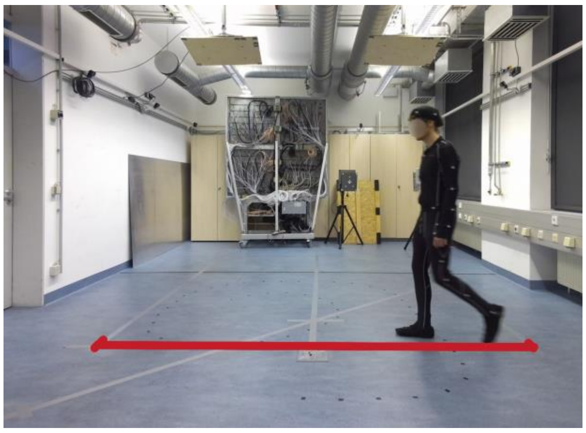
Walking 0° (W0)


The participant walks back and forth following a line parallel to the range axis.

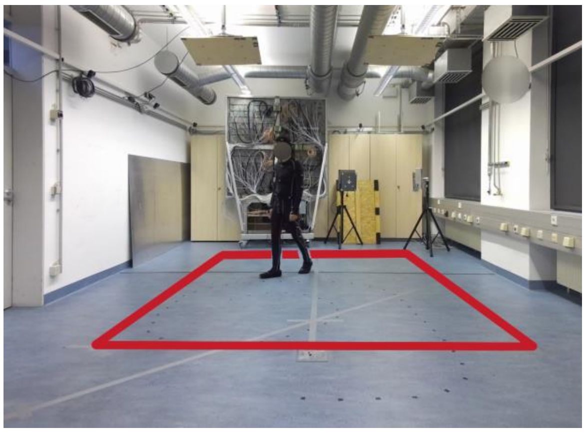
Walking 45° (W1)


The participant walks back and forth along the 45° line with respect to the range axis marked on the floor.

Walking 90° (W2)

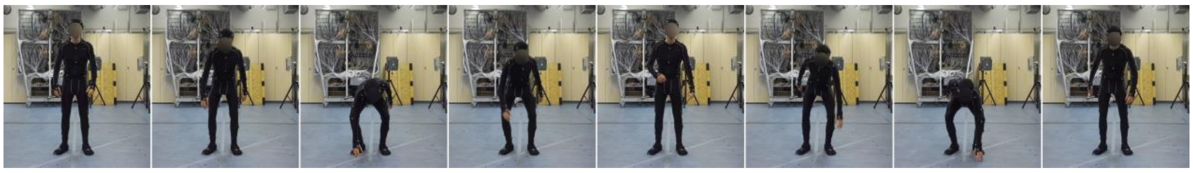
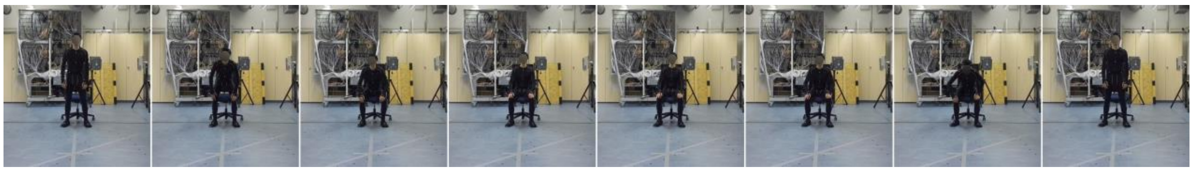
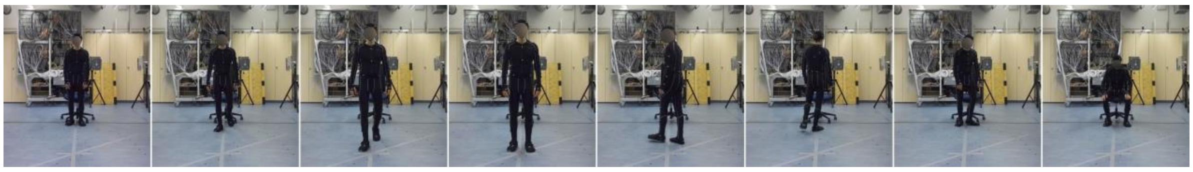
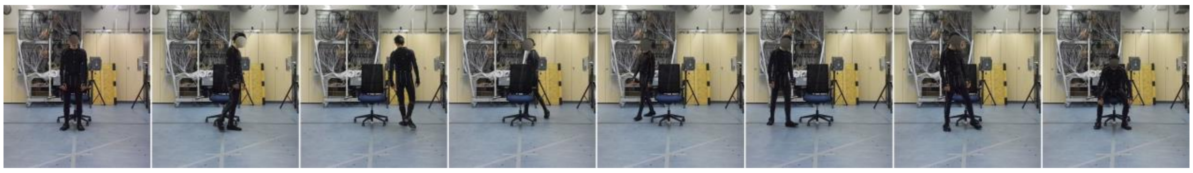
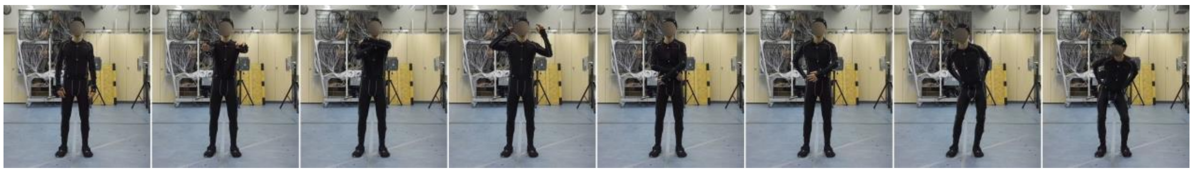

The participant walks back and forth over a line perpendicular to the range axis.

Random Walk (W3)


The participant walks freely inside the marked area.

Functional movements

These functional movements comprise some of the actions a healthy person must be able to perform in daily life. These movements are performed with the participant (and a chair in some cases) starting facing the radar.

Code	Action Name
F1	**Picking up**
	
The participant simulates the action of picking an object from the floor with a single hand. The participant slightly bends his torso and his knees, then he simulates picking an object. He picks the objects alternating hands.	
F2	**Standing up and sitting down**
	
The participant stands for approximately 2 seconds and sits on the chair for approximately 2 seconds.	
F3	**Standing up, waking and sitting**
	
The participant stands with the chair at his back, walks 2m and goes back to sit on the chair, stands up and repeats.	
F4	**Standing up, walking around the chair and sitting down**
	
The participant stands with the chair at his back, walks around the chair and sits back on the chair, stands up and repeats.	
F5	**Dancing (Macarena)**
	
The participant will dance the macarena and jump to turn 45°, repeat the dance and jump to turn 90° ending with a jump in place.	

Free movements

Finally, we will record free movements, where the participant can move without constraints in the marked area.

Code	Action Name
F6	**Boxing**
Boxing involves fluid and dynamic movements, incorporating punches and footwork that enable participants to navigate space freely.	

Last preparations and pos-processing

At the end of the recording session, the marker suit is retrieved from the participant.

The tracking data from Optitrack is saved.

The tracking data is saved and reviewed to correct mislabeled markers, interpolate missing markers if needed, etc. Once the data has been corrected it is saved as *.bvh* files and the joints information is retrieved in a *.csv*.

The radar pointclouds are obtained from the radar datacubes by using the CFAR algorithm.

The radar pointclouds and the joint's locations are linked together in a dataset.

.2 Apendix B

Participants Actions	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	Total
a0	151	128	151	186	0	221	231	241	151	161	171	171	1963
a1	411	495	497	477	503	496	484	486	502	496	490	898	6235
a2	437	514	501	493	498	511	495	480	511	491	500	482	5913
a3	445	525	524	483	503	507	498	499	508	506	0	478	5476
a4	429	518	476	478	512	515	497	505	507	499	489	484	5909
a5	438	523	515	505	513	532	524	520	509	509	444	505	6037
a6	416	514	498	501	499	516	472	930	496	510	484	487	6323
a7	468	1031	513	509	505	332	498	492	511	506	499	466	6330
a8	502	509	515	505	505	510	1006	472	510	513	509	473	6529
a9	430	518	502	518	513	516	501	456	509	442	496	391	5792
a10	0	514	508	514	515	521	511	503	513	510	497	498	5604
a11	0	513	516	509	515	515	501	513	505	503	505	476	5571
Total	4127	6302	5716	5678	5581	5692	6218	6097	5732	5646	5084	5809	67682

Table 1: Number of frames recorded within the action group 0, per participant and movement. The action group 0 was recorded with the participants facing the radar.

Participants Actions	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	Total
a0	151	151	210	199	211	381	211	191	171	171	171	166	2384
a1	510	532	417	508	497	520	493	488	0	509	495	498	5467
a2	512	523	513	507	513	514	488	494	508	507	500	501	6080
a3	512	511	513	511	516	501	480	492	514	515	499	495	6059
a4	469	489	495	496	489	514	496	467	496	507	458	435	5811
a5	520	518	520	510	511	535	519	501	517	515	513	510	6189
a6	508	500	499	505	506	516	496	472	509	514	496	499	6020
a7	462	0	516	507	509	512	493	488	513	510	510	486	5506
a8	507	0	516	498	509	477	499	467	510	511	506	498	5498
a9	504	518	514	469	515	497	503	951	500	509	492	492	6464
a10	517	516	514	503	512	512	505	504	515	503	466	387	5954
a11	518	512	468	498	507	494	508	503	515	508	456	440	5927
Total	5690	4770	5695	5711	5795	5973	5691	6018	5268	5779	5562	5407	67359

Table 2: Number of frames recorded within the action group 1, per participant and movement. The action group 1 was recorded with the participants rotated $45°$ with respect to the radar.

Participants / Actions	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	Total
a0	151	151	151	231	171	181	221	191	161	171	75	121	1976
a1	497	515	507	1003	506	490	494	914	510	508	503	502	6949
a2	472	509	513	506	0	492	492	489	510	495	504	504	5486
a3	502	509	519	506	485	490	492	490	516	514	474	503	6000
a4	470	492	501	473	503	486	497	499	498	505	497	468	5889
a5	502	515	524	514	512	502	507	518	531	505	473	511	6114
a6	491	513	516	510	994	485	511	497	505	504	495	500	6521
a7	438	506	521	513	501	498	506	508	436	512	506	475	5920
a8	478	512	513	515	506	494	502	474	504	502	506	483	5989
a9	469	502	482	504	505	497	499	485	509	511	416	457	5836
a10	492	514	0	506	493	496	489	465	477	512	504	490	5438
a11	509	962	518	508	502	441	486	509	509	508	461	509	6422
Total	5471	6200	5265	6289	5678	5552	5696	6039	5666	5747	5414	5523	68540

Table 3: Number of frames recorded within the actions action group 2, per participant and movement. The action group 2 was recorded with the participants rotated 90° with respect to the radar.

Appendix A

Acronyms

HPC	High Performance Computing
FAU	Friedrich-Alexander-Universität Erlangen-Nürnberg
NHR@FAU	Erlangen National High Performance Computing Center
DFG	German Research Foundation
mmWave	Millimeter wave
OMC	Optical Motion Capture
TI	Texas Instruments
RF	Radio Frequency
FMCW	frequency modulated continuous wave
MIMO	multiple-input multiple-output
EM	Electromagnetic
HPE	Human Pose Estimation
IMU	Inertial Measurement Unit
DNN	Deep Neural Network
CNN	Convolutional Neural Network

GNN Graph Neural Network

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

GRU Gated Recurrent Unit

MLP Multilayer Perceptron

FUSE Fast and Scalable Human Pose Estimation

MAE Mean Absolute Error

CFAR Constant False Alarm Rate

CA-CFAR Cell-Averaging Constant False Alarm Rate

CUT Cell Under Test

ML Machine Learning

AI Artificial Intelligence

DL Deep Learning

Lidar Light Detection and Ranging

Mars mmWave-based Assistive Rehabilitation System for Smart Healthcare

CSI channel state information

ReLU rectified linear unit

ELU exponential linear unit

SELU scaled exponential linear unit

GELU Gaussian-error linear unit

SGD stochastic gradient descent

FFT fast Fourier transform

PRI pulse repetition interval

MSE	Mean Squared Error
PCK	Percentage of Correct Keypoints
MPJPE	Mean Per Joint Position Error
VCO	Voltage Controlled Oscillator