Integrating Symmetries into Skeleton-based Action Recognition

Master Thesis

presented by
Laura Katharina Prasse
Matriculation Number 1721782

submitted to the
Data and Web Science Group
Prof. Dr. Margret Keuper
University of Mannheim

November 2022

Abstract

This is where I write my abstract.

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

Humans can easily distinguish between observed actions. Occlusion or suboptimal lighting have a minor effect on the recognition capabilities. For machines however, this poses a major challenge. It is a current research interest to change this.

Video surveillance systems can benefit from increased automation, such that certain actions trigger pre-defined reactions. Examples may be the notification of the police in case of criminal activity or the sending of emergency calls in case of an accident or a person's critical condition. Further applications can be found in the field of human computer interaction, video retrieval and video indexing. The further one likes to think forward, the more applications become imaginable, assuming that machines will have an ever-growing presence in our lives. The inclusion of activity recognition requirements in the context of automated vehicles into European law took place already in 2019, and entered into force recently, on July 6, 2022 [4]. This law requires interior cameras to monitor the driver's activities and to determine, after how many seconds the human driver can take over the control from the autopilot. This further illustrates the need for advancement in this field, as this technology is currently needed.

Cameras are required to record the action before the computer can try to recognize it and recognition itself is a classification task. Its difficulty depends on the similarity of the input actions, e.g. standing up and sitting down contains similar movements and can only correctly be recognized based on the order of movements. The research on action recognition is an active field since the late 1990s [8].

First advances into the field used Recurrent Neural Networks [RNNs] such as Long Short-Term Memory [LSTM] with the aim to capture the action over a period of time [3, 20, 17]. Later CNNs were used, for which the action was depicted as a 2D map which included e.g. joint type, time stamp and position

[5, 18, 15, 7]. In order to recognize the action, a large receptive field is necessary which causes high model complexity [19]. In recent years, Graph Convolutional Networks [GCN] show the highest performance in terms of action recognition accuracy [12, 16, 11, 1]. In contrast to CNNs, GCNs can use the inherent structure of the skeleton data. The joints are represented by nodes in the graph and the edges stand for the connection of the joints [11] or the correlation of joints [1].

Action recognition uses various formats of input data, ranging from RGB video frames to skeletons extracted from videos. A skeleton consists of 2D or 3D coordinate data for human key joints, e.g. shoulder, hip and knee. This information can be enriched by additional data such as velocity [19]. Using skeleton data instead of video data does not only increase efficiency but also enhance robustness against variation of viewpoint and appearance [1, 11, 19].

Studies differ in the way they utilize the spatial and temporal information when recognizing the action. Several studies first extract spatial and then temporal information [19, 1]. While Zhang et al. accesses both types of information once, Chen et al. iterate over the input data several times, both accessing the spatial and temporal data. Zhang et al.'s model Semantics-Guided Neural Network [SGN] assigns semantic labels to both joints and skeletons, joint type and time stamp respectively, which they feed into the neural network as additional joint information, next to joint position and velocity [19]. While both studies show a comparable performance on the simpler NTU-60 dataset, the SGN model's performance is ten percentage points below Chen's Channel-wise topology refinement Graph Convolutional Network [CTR-GCN] on the NTU-120 dataset.

1.1 Problem Statement

Current state-of-the-art models achieve a accuracy of up to 90% [1]. This suffices for some applications, such as video surveillance where any automation is an improvement compared to the status quo. Furthermore in the surveillance context, manual reviewing of the action recognition is possible. Given the context of autonomous driving or human-robot interaction, this is not the case. To continue along the example of self-driving cars: they are continuously recognizing the environment around them, in particular object and action recognition. While object recognition has advanced to the point that self-driving cars are allowed to drive in the US, action recognition is incorrect for every tenth action.

The purpose of this master thesis is to improve that. The NTU120 dataset is the most challenging benchmark dataset currently available to my knowledge and used for this analysis [6]. When reviewing the actions, it becomes apparent that many of them are symmetric. This includes both symmetry over time and over joint.

Symmetry over time can be observed in the action e.g. ”hand waving” [A23], ”rub two hands together” [A34], ”juggling table tennis balls” [A66]. In these cases, one or multiple joints repeat the same motion repeatedly over time. Symmetry over joint is inherent to the actions ”sitting down” [A8], where e.g. both knee joints have the same motion at each time step. Further actions with this type of symmetry are e.g. ”standing up (from sitting position)” [A9], ”put the palms together” [A39], and ”blow nose” [A105].

An existing baseline model is used to determine the effectiveness of modifications or alternatively, the addition of a symmetry module. It is investigated, if integrating symmetry can improve the accuracy of the baseline model. Both types and their combination are analysed and their effectiveness is discussed.

1.2 Contribution

Chapter 2

Theoretical Framework

2.1 State-of-the-Art

[11] [19] [1]

2.2 Frequency-based methods

Action MACH Filter are proposed by Rodriguez, Ahmed, and Shah, which employ Maximum Average Correlation Height Filter. This method is explained closely following the authors' publication [9]. For each action class a filter is generated based on the training data. For efficiency reasons, these calculations are performed in the frequency domain. The action filter has three components, noise, power spectra, and similarity, which can be regulated by parameters. To detect an action the normalized correlation of the Action MACH applied to a test video is assessed and evaluated in comparison to a learned threshold for each action. The advantages of this method are robustness against intra-class variance and efficiency.

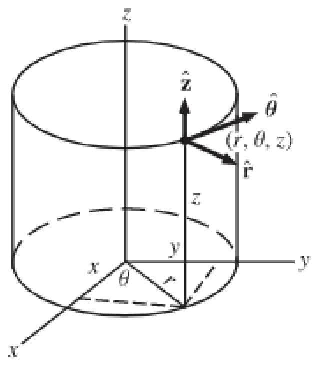
The input data consists of action videos, transformed into their temporal derivatives. The 3-D Fourier transform is computed and the output is a spatiotemporal volume in frequency domain. Based on this, a single column vector is generated by concatenating the three dimensions. For each action, the column vectors of all samples are stacked into a matrix. This matrix is used to generate the Action MACH as follows from equation 2.1

$$h = (\alpha C + \beta D_x + \gamma S_x)^{-1} m_x \tag{2.1}$$

where m_x is a vector containing the mean of all samples, a row vector. C is representing the noise covariance matrix, D the average power spectral density, and S is the average similarity matrix. The parameters α, β, and γ adjust the sensitivity of the filter to the three measures. Lastly, the 1D filter is stacked into a 3D one by repeating the initial transformation in reversed direction and transformed back from frequency to the spatial domain.

Motion History Volumes are presented by Weinland, Ronfard, and Boyer as alternative approach to recognizing action. Their approach is described closely following their research [14]. This method transforms the recorded action into motion history volumes which depict the history of motion occurrence in 4D. The motion representation in Cartesian coordinates (x, y, z) is transformed into cylindrical coordinates (r, θ, z) as shown in figure 2.1

Figure 2.1: Visualization of Cylindrical Coordinates



where r represents the radial coordinate, θ the azimuth and the height z remains unchanged. Equations 2.2 include the formulae for the conversion between the two coordinate systems.

$$r = \sqrt{(x^2 + y^2)} \tag{2.2a}$$

$$\theta = \tan^{-1} \frac{y}{x} \tag{2.2b}$$

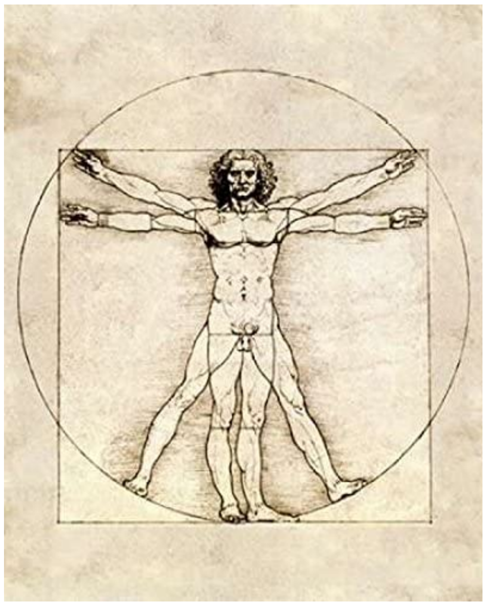
$$z = z \tag{2.2c}$$

Subsequently, the coordinate data is transformed in the frequency domain using a 1D FFT with respect to the azimuth. The magnitude of Fourier transform is analysed to detect the action by classification. The result is normalized in regard to motion length. The authors report the best results when reducing dimensionality by Principal Component Analysis [PCA] and Linear Discriminant Analysis [LDA]. Advantages of this approach are rotation invariance and robustness to noise.

Actionlets Wang et al. employ actionlets to recognize actions [13] .

2.3 Spherical Coordinates and Harmonics

Figure 2.2: Visualisation of joints in NTU120 dataset as published by Liu et al.



Chapter 3

Methodology

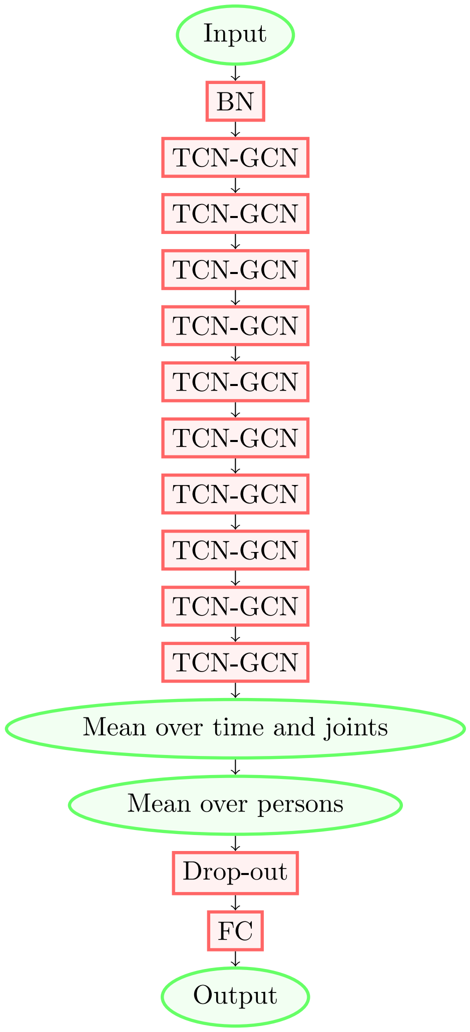
A human skeleton can be represented as a graph by denoting its joints as vertices and the *bones* connecting the joints as edges. The human skeleton is thus represented by the graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, containing the vertice set $\mathcal{V} = \{v_1, v_2, .., v_N\}$ and the edge set $\mathcal{E} = \mathbf{A} \in \mathbb{R}^{NxN}$, which is the adjacency matrix of the graph. The matrices are depicting the correlations between the vertices, which do not have to be symmetrical (for further details compare the general section of Chapter **??**). The joints' features are captured by $\mathcal{X} \in \mathbb{R}^{NxC}$, which contains a C dimensional feature for each vertex.

3.1 Baseline Model

This study uses the baseline model provided by Chen et al. [1]. While their study proposes a channel-wise topology refinement model, their baseline uses a topology-shared method, i.e. the topology is shared across all channels. Furthermore, their model uses a static method, where the adjacency matrix is a parameter and learned during training and not generated based on the inputs. The adjacency matrix is initialized as a identity matrix. In the following section, the model is described in detail closely following Chen et al. [1].

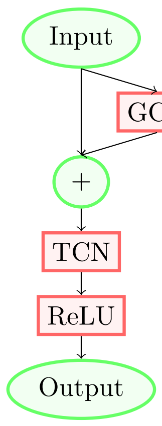
The baseline model accepts three inputs: 3D joint coordinates, 3D bone vectors, and 3D joint velocities. All types of inputs have 3 channels, which are first embedded into 64 dimensions (layer 1 - 5), then 128 dimensions (layer 6 - 8), and finally 256 dimensions (layer 9 - 10). Figure 3.1 visualizes the architecture of the baseline model. Its main components are the TCN-GCN units, which contain a spatial unit and a temporal unit. After 10 TCN-GCN units, the data is aggregated, passed through a drop-out layer and classified by a fully connected layer into the respective number of output classes.

Figure 3.1: Visualization of a TCN-GCN Baseline Model



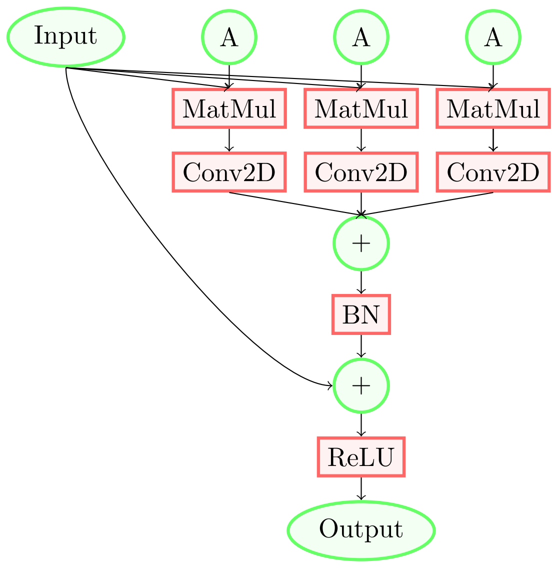
All TCN-GCN units have the same structures, as shown in Figure 3.2. They each consist of one spatial GCN unit and one temporal TCN unit. The input data is fed into the GCN unit and thereafter added to an unmodified version of the input. In layer 1, 6, and 9 where the GCN increases the dimensionality of the data, the input is fed through an additional TCN unit in order to have the number of dimensions required for the addition. Subsequently, the data is fed into the temporal TCN unit, in which the change over time is embedded. The last component is a Rectified Linear Unit [ReLU] which forces the data to remain positive.

Figure 3.2: Visualization of a TCN-GCN Unit



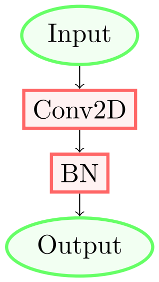
The spatial GCN unit is visualized in Figure 3.3. It contains the matrix multiplications of the input with each of the three adjacency matrices separately. Each adjacency matrix is embedded using a 1x1 kernel. Subsequently, the three adjacency matrices are added together and batch normalized. Thereafter, the data is added to an unmodified version of the input. Unmodified may include the alteration of dimensionality, in layers 1,6 and 9, where a convolution and batch norm is computed. Finally, a ReLU activation function is applied.

Figure 3.3: Visualization of a GCN Unit



The temporal TCN unit is depicted in Figure 3.4 where five succeeding time steps are convolved. This is identical across all layers, however the stride differs. Mainly a stride of 1 is used, expect for in layer 5 and 8 where a stride 2 is chosen. After the convolution, the data is batch normalized.

Figure 3.4: Visualization of a TCN Unit



3.2 Baseline Modifications

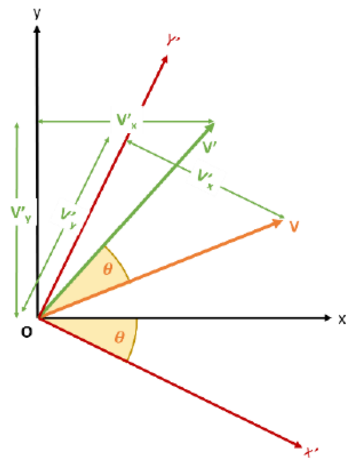
3.3 Skeleton Normalization

One approach to detecting symmetries is the normalization of the input data in a way that makes recurring motions more apparent. With this aim, the skeleton's spine was rotated so that it lies on the z-axis. Numerically, this increases the comparability of arm or leg motion e.g. the identical motion of both arms would only differ by sign, given the joint recording and detection is of high accuracy. Even with noisy data, this normalization is expected to improve the recognizability of joint-wise symmetry. This section describes the normalization procedure, whereof the subsections 3.3.1 and 3.3.2 are written in close alignment with Cole [2].

3.3.1 Rotation Matrix Derivation

All rotations are defined by an angle θ and an axis. Vector rotations are by definition the counter-clockwise rotation of a vector $\mathbf{v}$. The angle θ can be measured between the original vector $\mathbf{v}$ and its rotated version $\mathbf{v}'$. Alternatively, the same angle θ is obtained when rotating the coordinate axis clockwise, as visualized in Figure 3.5. In $\mathbb{R}^3$, the three-dimensional Euclidean space, all axes can be used for rotation. The coordinate corresponding to the rotation axis is invariant, i.e. remains unchanged by the rotation, e.g. when z is the rotation axis, $v_z = v'_z$. Rotations in $\mathbb{R}^2$ can be imagined as selecting the z-axis as rotation axis.

Figure 3.5: Equivalence of counter-clockwise rotated vector and clockwise rotated coordinate system



A vector rotation is mathematically identical to multiplying a vector from the right with a rotation matrix $\mathbf{R}$, as visualized in equation 3.1. The rotated vector has the same dimensionality as the original vector, i.e. $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^{3\times1}$, thus requiring the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$. Given the fact that the vector component corresponding to the rotation axis is invariant, this part of the rotation matrix that $\mathbf{v}$ is multiplied with must be the identity, as $\mathbf{I}\mathbf{v} = \mathbf{v}$.

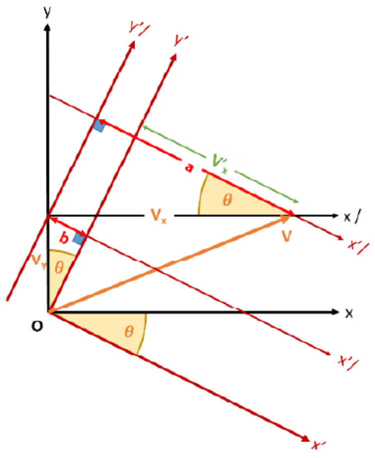
$$\mathbf{R}\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} \tag{3.1}$$

This use case requires the rotation onto a given axis, the z-axis. At first, a vector representing the spine is derived from the data and normalized to unit length. Secondly, it is rotated from an arbitrary position in $\mathbb{R}^3$ onto the plane spanned by any two axes, resulting in the vector corresponding the the remaining axis is zero. For example, if the vector is rotated onto the xz-plane then the y-coordinate of the vector is be equal to zero. Lastly, the vector is rotated along another axis until it is aligned with the desired axis of the coordinate system, in this case, the z-axis. Rotation matrices for all three axis are given in Equation 3.2, each including parts of an identity matrix, as explained above.

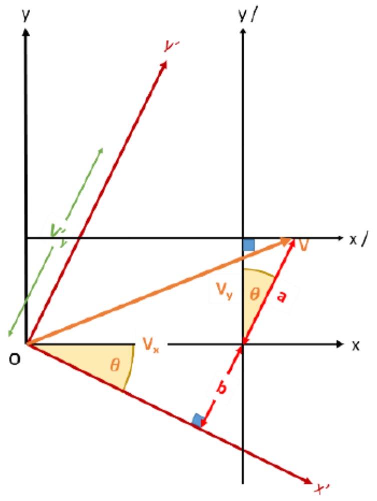
$$
\begin{gathered}
\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \mathbf{R_y} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \\
\mathbf{R_z} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{gathered}
\tag{3.2}
$$

In order to obtain the rotation matrices, the angle θ has to be determined. This can be achieved by using linear algebra, more precisely trigonometry. As explained above, rotations always take place along two axis at a time. When the rotation is a along the z-axis, the x and the y-axis need to be considered. Figure 3.6a visualizes how to compute the x-coordinate of the rotated vector,$\mathbf{v'_x}$.

Figure 3.6: Rotation Matrix Derivation, when z is the rotation axis.



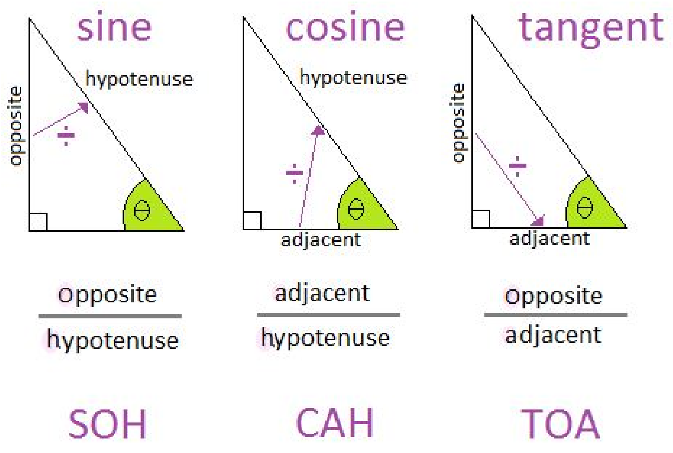
(a) x-coordinate



(b) y-coordinate

From the coordinate system, some measurements are know while others can be inferred. The value v_x is known and can be used to construct a right triangles with v_x, the parallel of rotated y-axis y' that passes through the origine, and the parallel of the rotated x-axis x' passing the two other vectors. When computing angles, any parallel of the vectors can be used, as they all have identical angles. The parallel of the y-axis is chosen such that the full length of v_x can be used, without requiring additional computation. This triangle is a right triangle because the x-axis and the y-axis are perpendicular to each other. This triangle can be used to compute the value of a, which contains the v'_x value. Using the trigonometry results in Figure 3.7, the value a can be computed using the cosine, i.e. $\cos\theta = \frac{adjacent}{hypotenuse}$. In this case, the hypotenuse is v_x and theta is the angle of rotation, since its the angle between the original and the rotated x-axis. This $adjacent = \cos\theta * hypotenuse$, in this case $a = v_x * \cos\theta$.

Figure 3.7: Overview of Trigonometric Rules



Similarly, a second right triangle can be constructed to compute b, such that $v'_x = a - b$. It contains the original and the rotated y-axis, in-between which the angle θ can be found. The side lying on the y-axis has the known length of v_y, and its third side is the parallel of the rotated x-axis through v_y. The rotated x and y-axis are perpendicular, allowing to compute b using the sine, i.e. $\sin\theta = \frac{opposite}{hypotenuse}$ rearranged into $opposite = \sin\theta * hypotenuse$. $b = v_y * \sin\theta$.

The coordinates of the y-coordinate can be computed in a similar fashion, as visualized in Figure 3.6b. In summary, the x-coordinate of v' is equal to $v_x * \cos\theta - v_y * \sin\theta$ and the y-coordinate equals to $v_x * \sin\theta + v_y * \cos\theta$. Using these computation, the rotation matrix can be constructed, in which the first column contains all elements corresponding the the x-coordinate, the second row containing the ones for the y-coordinate, as shown in Equation 3.2.

3.3.2 Skeleton Rotation

The angle between two vectors can be computed using the dot product, as shown in Equation 3.3. This method is used to compute the rotation angle θ.

$$\cos\theta = \frac{\vec{v'} \cdot \vec{v}}{\left|\vec{v'}\right| * |\vec{v}|} \tag{3.3}$$

For this use case, the angle needs to be computed under consideration of the invariance of the coordinate corresponding the the rotation axis. Continuing with the example above, where the z-axis was the rotation axis, two vectors need to be constructed originating at the invariant point v_z = $\left[\begin{smallmatrix} 0 \\ 0 \\ v_z \end{smallmatrix}\right]$ and pointing towards v and v' respectively, as displayed in Equation 3.4. Since the rotation is independent of z, $v_z = v'_z$, and since the vector v' lies in the xz-plane, its y-coordinate has to be zero.

$$\mathbf{v}' = \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ v_z \end{bmatrix} = \begin{bmatrix} v'_x \\ v'_y \\ 0 \end{bmatrix} = \begin{bmatrix} v'_x \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ v_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix} \tag{3.4}$$

The vectors v and v' are plugged into Equation 3.3, allowing to compute $\cos\theta$, as shown in Equation 3.5.

$$\begin{aligned} \cos\theta &= \frac{\left[\begin{smallmatrix} v'_x \\ 0 \\ 0 \end{smallmatrix}\right] \cdot \left[\begin{smallmatrix} v_x \\ v_y \\ 0 \end{smallmatrix}\right]}{\sqrt{\left[\begin{smallmatrix} v'_x \\ 0 \\ 0 \end{smallmatrix}\right]^2} * \sqrt{\left[\begin{smallmatrix} v_x \\ v_y \\ 0 \end{smallmatrix}\right]^2}} \\ &= \frac{v'_x * v_x}{\sqrt{{v'_x}^2} * \sqrt{v_x^2 + v_y^2}} = \frac{v'_x * v_x}{v'_x * \sqrt{v_x^2 + v_y^2}} = \frac{v_x}{\sqrt{v_x^2 + v_y^2}} \end{aligned} \tag{3.5}$$

The rules of trigonometry $\cos^2\theta + \sin^2\theta = 1$, which can be re-written and simplified as presented in Equation 3.6. Due to the fact, that the y-coordinate of the rotated vector has be zero, the sign must be negative. The values for $\sin\theta$ and $\cos\theta$ are then plugged into the rotation matrix. The derivation for a rotation around the z-axis and onto the xz-plane has been described in detail; rotations around the x or y-axis can be derived in the same manner.

$$\begin{aligned}
\sin\theta &= \pm\sqrt{1-\cos^2\theta} \\
&= \pm\sqrt{1-\left(\frac{v_x}{\sqrt{v_x^2+v_y^2}}\right)^2} = \pm\sqrt{1-\frac{v_x^2}{v_x^2+v_y^2}} \\
&= \pm\frac{v_y}{\sqrt{v_x^2+v_y^2}}
\end{aligned} \tag{3.6}$$

3.4 Alternative Coordinate Systems

An alternative approach to the normalization of the coordinates described in section 3.3, is the conversion of the coordinate system from the Cartesian to an alternative one. The literature indicates that cylindrical and spherical coordinates can improve the detectability of repetitive actions, as described in chapter 2.

For the conversion from between the coordinate systems, formulae exist which translates the Cartesian coordinates (x, y, z) into spherical coordinates (ρ, θ, ϕ) or cylindrical coordinates (r, θ, z). Two solutions exist, how the conversion can be implemented; local and global conversion are both described and evaluated against each other in the subsequent section. The general method is identical for spherical and cylindrical coordinates; while the subsequent example focuses on spherical coordinates, the same method is applied for cylindrical coordinates with slightly modified formulae.

3.4.1 Global Conversion

The first approach is to compute the spherical coordinates for each skeleton, thus utilizing a global conversion. The coordinate values (ρ, θ, ϕ) are optimally computed independent of the orientation of the skeleton. This can be achieved by either using background knowledge or formulae.

Background Knowledge

Firstly, the knowledge of the human skeleton can be used to calculate the angles or the distance in accordance to the body orientation. The azimuth θ is the angle between the spine and the newly constructed vector connecting the base of the spine to each joint. Further, the colatitude ϕ is the angle between the hip and the newly constructed vector connecting a hip bone to each joint. This measurement depends on the hip bone chosen as starting point for both vectors. Here it is important to consistently chose the same bone as starting point for both vectors, i.e. hip vector and hip-joint vector. The radial distance ρ is simply the length of the initial vector. During preprocessing, the dataset is centered at the origine in a way that the second joint, middle of spine, has the Cartesian coordinates **O** = $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$.

Algorithm 1 Global Conversion to Spherical Coordinates using Background Knowledge

```
Input: $x_{cart}$: Skeleton joints 3D Cartesian coordinates, centered
Output: $x_{sph}$: Skeleton joints 3D spherical coordinates
     spine = joint$_{spineatshoulderlevel} - joint_{baseofspine}$
spine$_{norm}$ =spine$\frac{}{spine_2}$
hip = joint$_{lefthipbone} - joint_{righthipbone}$
$\rho$ = $\frac{v_x}{\sqrt{v_x^2+v_y^2}}$
0: for all $i \in T$ do
0:     $amt1, date1 \leftarrow T_i$
0:     for all $j \in i+1$ do
0:         $amt2, date2 \leftarrow T_j$
0:         if MatchReview($amt1, date1, amt2, date2$) then SaveResultToSql($V_{cand}$,$A$)
0:         end if
0:     end for
0: end for
0: return $V_{cand}$, $A$ =0
```

$\cos\theta$ and θ are have a linear relationship for most parts, thus $\cos\theta$ is used. Basically formula for cosine similarity. Identical vectors have a similarity of 1, 90 degree angle have a similarity of 0. vectors pointing in exactly opposite directions have to similarity -1.

’s Pythagoras theorem, i.e. $a^2 + b^2 = c^2$

3.5 Symmetry Module

Chapter 4

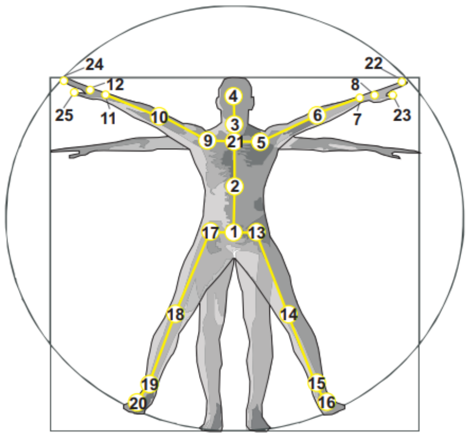
Experiments

4.1 Dataset

The benchmark dataset NTU RGB+D 120 [6] is used to evaluate the effectiveness of the proposed modifications. It contains 114,480 videos featuring 106 distinct subjects from 15 different countries [6]. The subject's age ranges between 10 and 57 years and their height varies between 130 cm and 190 cm [6]. Both sexes are included in the dataset [6]. For all demographic data, no statistics are published and thus no information is available about the distribution of the data.

The dataset was generated using Microsoft Kinect v2 sensors, which recorded the actions in four different data modalities. The authors provide RGB video frames, depth maps, 3D joint information, and infrared (IR) sequences [6]. As shown in Figure **??**, the joint data includes the 25 main joints, namely (1) base of spine, (2) middle of spine, (3) neck, (4) head, (5) left shoulder, (6) left elbow, (7) left wrist, (8) left hand, (9) right shoulder, (10) right elbow, (11) right wrist, (12) right hand, (13) left hip, (14) left knee, (15) left ankle, (16) left foot, (17) right hip, (18) right knee, (19) right ankle, (20) right foot, (21) spine, (22) tip of left hand, (23) left thumb, (24) tip of right hand, (25) right thumb [6].

Figure 4.1: Visualisation of joints in NTU120 dataset as published by Liu et al. [6]



The dataset includes recordings from 155 distinct camera viewpoints [6]. In total, the benchmark contains 32 unique set-ups, differing in camera height and distance to the action performing subject(s). For each recording, a set-up and one of 96 different backgrounds with varying illuminations is chosen. Each subject is acted to perform each action two times. While performing, the subject is recorded by three cameras positioned according the selected scenario. While the cameras' heights are identical, their angles towards the subject differ from full frontal ($0°$), over $45°$-view, to side-view ($90°$). This results in 6 recordings for each action performance: two full frontal-views, one left and one right $45°$-view and one left and one right side-view.

The recorded actions can be categorized into three groups: daily actions, health-related actions and mutual actions. While the first two categories only require a single subject, the third one requires two subjects. Table 4.1 displays the categories and provides examples for each of them, as published by Lui et al. [6].

Action Category	#	Examples
Daily	82	Eating, writing, sitting down, moving objects
Health-related	12	Blowing nose, vomiting, staggering, falling down
Mutual	26	Handshaking, pushing, hitting, hugging

Table 4.1: Action categories of benchmark dataset NTU RGB+D 120

The category mutual actions contains several violent and potentially crimilal actions such as ”hit other person”, ”wield knife towards other person”, or ”shoot at other person with a gun".

To my knowledge, the NTU RGB+D 120 dataset is the most extensive and challenging benchmark available. The authors included fine-grained motions, objects and similar actions in various combination to achieve this. Examples for fine-grained motions without objects are "make ok sign" and "make victory sign". Furthermore, fine-grained individual actions with objects e.g. "count money" and "play with Rubik's cube" and fine-grained mutual actions with objects e.g. "wield knife towards other person" or "hit other person with object". The similar actions include "touch other persons pocket (steal)" and "grab other persons stuff", which contain similar motions executed at different speeds. Other similar actions include the same motion with the different objects e.g. "put on jacket" and "put on bag/backpack".

The benchmark dataset can be evaluated in two settings: the cross-subject and the cross-setup evaluation. In the cross-subject evaluation, the 106 subjects are split equally into two groups containing each 53 subjects whereof one is used exclusively for training and the other exclusively for testing. In the cross-setup evaluations, the 32 scenarios (camera height and distance to subject) are split equally into two groups of 16, which are used either exclusively for training or testing.

According to Liu et al., both individual and mutual actions including distinct motions can be well recognized based on the joint information. Further, they report that similar actions, both with and without objects are hard to distinguish e.g. "take a photo of other person" and "shoot at other person with a gun".

4.2 Preprocessing

The baseline model employed the preprocessing proposed by Zhang et al. [19], which is described in detail in the following section, following closely the original paper.

The skeleton data is loaded and [10] bone information as vector from origin joint to target joint. Each body's data is a dict that contains the following keys: - joints: raw 3D joints positions. Shape: $(\mathrm{num}_frames x 25, 3) - colors : raw 2D color locations. Shape : (num_frames, 25, 2) - interval : a list which stores the frame indices of this$ $motion : motion amount (only for the sequence with 2 or more body IDs)$.

Return: a dict for a skeleton sequence with 3 key-value pairs: - name: the skeleton filename. - data: a dict which stores raw data of each body. - num_frames: the number of valid frames.

The joint "spine" which is located at the middle of the spine was aligned with the origine $\mathbf{O} = [0, 0, 0]$.

4.3 Settings

Chapter 5

Results

#	Modification	Layer [unit]	Parameters	Accuracy	Change (%)
BL	-	-	*2,108,322*	*0.8375*	-
1	3D velocity added	1 [T]	2,108,322	**0.8439**	**+0.6%***
BL	*Double number of input channels*	*1 [T]*	*2,128,802*	*0.8378*	-
2	3D velocity concat.	1 [T]	2,128,802	0.8406	**+0.3%**
BL	-	*1 [T]*	*2,128,482*	*0.8343*	-
3	Embed. FFT mag. [1:] of 3D coordinates concat.	1 [T]	2,128,482	**0.8408**	**+0.7%**

Table 5.1: Overview of selected experimental results of the CTR-GCN baseline model evaluated on NTU-120 XSub

5.1 CTR-GCN

XSet: fourier 6c 0.8584, baseline 0.8521

#	Modification	Layer [unit]	Parameters	Accuracy	Change (%)
BL	-	-	*2,108,322*	*0.8375*	-
1	3D velocity added	1-10 [G]	2,108,322	0.8376	± 0%
2	3D velocity & cum. velocity over joint added	1-10 [G]	2,108,322	0.7968	-4.1%
3	3D cum. velocity over joint added	1-10 [G]	2,108,322	0.8021	-3.5%
4	3D velocity & cum. velocity over time added	1-10 [G]	2,108,322	0.8316	-3.5%
5	3D cum. velocity over time added	1-10 [G]	2,108,322	0.8309	-0.6%
6	3D cum. velocity over joint added	1-10 [G]	2,108,322	0.8021	-3.5%
7	3D velocity added	1 [G]	2,108,322	0.8363	-0.1%
8	3D velocity added	5 [G]	2,108,322	0.8357	-0.2%
BL	-	*1 [G]*	*2,108,578*	*0.8395*	-
9	1D magnitude of velocity concat.	1 [G]	2,108,578	0.8376	-0.2%
BL	-	-	*2,108,322*	*0.8375*	-
10	3D velocity added	1-10 [T]	2,108,322	0.8314	-0.6%
11	3D velocity added	1 [T]	2,108,322	**0.8439**	**+0.6%***
12	3D cum. velocity over time added	1 [T]	2,108,322	0.8388	**+0.1%**
BL	*Double number of input channels*	*1 [T]*	*2,128,802*	*0.8378*	-
13	3D velocity concat.	1 [T]	2,128,802	0.8406	**+0.3%**

Table 5.2: Velocity in the CTR-GCN baseline model evaluated on NTU-120 XSub

5.2 SGN

5.3 Normalization

#	Modification	Layer [unit]	Adjacency Embed.	Parameters	Accuracy	Change (%)
BL	-	-	*3*	*2,108,322*	*0.8375*	-
1	3D velocity added	1 [T]	3	2,108,322	**0.8439**	**+0.6%***
BL	-	-	*6*	*2,783,136*	*0.8425*	-
2	3D velocity added	1 [G]	6	2,783,136	**0.8433**	**+0.1%**
3	3D cum. velocity o. time added	1 [G]	6	2,783,136	0.8311	-1.1%
4	3D cum. velocity o. joint added	1 [G]	6	2,783,136	0.8393	-0.3%
5	3D velocity added	1 [T]	6	2,783,136	**0.8455**	**+0.3%**
6	3D cum. velocity o. time added	1 [T]	6	2,783,136	0.8414	-0.1%
7	3D cum. velocity o. joint added	1 [T]	6	2,783,136	0.8327	-1.0%
BL	*Double number of input channels*	*1 [G]*	*6*	*2,784,480*	*0.8397*	-
8	3D velocity concat.	1 [G]	6	2,784,480	**0.8420**	**+0.2%**
9	3D cum. velocity o. time concat.	1 [G]	6	2,784,480	**0.8421**	**+0.2%**
10	3D cum. velocity o. joint concat.	1 [G]	6	2,784,480	**0.8422**	**+0.3%**
BL	*Double number of input channels*	*1 [T]*	*6*	*2,803,616*	*0.8407*	-
11	3D velocity concat.	1 [T]	6	2,803,616	**0.8448**	**+0.4%**
BL	-	-	*4*	*2,333,580*	*0.8361*	-
12	1D magnitude of velocity concat.	1 [G]	4	2,333,580	0.8401	**+0.4%**
BL	-	-	*4*	*2,333,580*	*0.8379*	-
13	1D magnitude of velocity concat.	1 [T]	4	2,333,580	0.8377	±0%
BL	*Double number of input channels*	*1 [G]*	*6*	*2,784,480*	*0.8397*	-
12	FFT embed. dim. of 3D coordinates concat.	1 [G]	6	2,784,480	0.8404	+0.1%
13	FFT embed. dim. of 3D velocity concat.	1 [G]	6	2,784,480	0.8378	-0.2%
14	FFT embed. dim. of 3D cum. velocity over joint concat.	1 [G]	6	2,784,480	0.8338	-0.6%

Table 5.3: Analysis of adjacency embeddings (1.Level): Velocity and frequency representation in the CTR-GCN baseline model evaluated on NTU-120 XSub

#	Modification	Layer [unit]	Parameters	Accuracy	Change (%)
BL	-	-	*2,108,322*	*0.8375*	-
1	FFT mag. [:] of 3D coordinates instead of input	1 [G]	2,108,322	0.8259	-1.2%
2	FFT mag. [:] of 3D coordinates added	1 [G]	2,108,322	0.8256	-1.2%
8	FFT mag. [1:] of 3D velocity added, zero vector prepended	1 [T]	2,108,322	0.8351	-0.2%
BL	-	*1 [G]*	*2,109,090*	*0.8371*	-
3	FFT mag. [:] of 3D coordinates concat.	1 [G]	2,109,090	**0.8383**	**+0.1%**
BL	-	*1 [T]*	*2,128,802*	*0.8378*	-
4	FFT mag. [:] of 3D coordinates concat.	1 [T]	2,128,802	**0.8392**	**+0.1%**
BL	-	*1 [T]*	*2,118,882*	*0.8393*	-
5	RFFT mag. [:] of 3D coordinates concat.	1 [T]	2,118,882	**0.8400**	**+0.1%**
BL	-	*1 [T]*	*2,128,482*	*0.8343*	-
6	FFT mag. [1:] of 3D coordinates concat.	1 [T]	2,128,482	**0.8408**	**+0.7%**
7	FFT phase [1:] of 3D coordinates concat.	1 [T]	2,128,482	**0.8370**	**+0.3%**
8	FFT mag. [1:] of 3D coordinates plus velocity concat.	1 [T]	2,128,482	**0.8408**	**+0.7%**
9	FFT phase [1:] of 3D coordinates plus velocity concat.	1 [T]	2,128,482	**0.8370**	**+0.3%**
BL	*Bone vector*	*1 [T]*	*2,128,482*	*0.8482*	-
10	FFT mag. [1:] of bone vector concat.	1 [T]	2,128,482	**0.8480**	**±0%**
11	FFT phase [1:] of bone vector concat.	1 [T]	2,128,482	**tbd**	**??%**
BL	*Velocity*	*1 [T]*	*2,128,482*	*0.8036*	-
12	FFT mag. [1:] of velocity concat.	1 [T]	2,128,482	0.8040	±0%
13	FFT phase [1:] of velocity concat.	1 [T]	2,128,482	0.8037	±0%

Table 5.4: Frequency based methods in the CTR-GCN baseline model evaluated on NTU-120 XSub, fourier transform over embedding dimension

#	Modification	Param.	Accuracy	Change (%)
BL	-	*2,118,562*	*0.8396*	-
1	RFFT mag. [1:] of 3D coordinates concat.	2,118,562	**0.8429***	**+0.3%**
2	RFFT mag. [1:] of 3D coordinates concat., normalize by $1/n$	2,118,562	0.8388	-0.1%
3	RFFT mag. [1:] of 3D coordinates concat., normalize by $1/\sqrt{n}$	2,118,562	**0.8408**	**+0.1%**
4	RFFT mag. [1:] of 3D coordinates concat., normalize by $\log x + 0.01$	2,118,562	0.8302	-0.9%
5	RFFT mag. [1:] of 3D coordinates concat., normalize by first coefficient	2,118,562	0.8293	-1.0%
6	RFFT2 mag. [1:] of 3D coordinates concat., indep. of time	2,118,562	0.8180	-2.2%
7	RFFT2 mag. [1:] of 3D coordinates concat., indep. of joint	2,118,562	0.7539	-8.6%
9	RFFT mag. [1:] of 3D velocity concat.	2,118,562	0.8348	-0.5%
BL	-	*2,118,242*	*0.8376*	-
10	RFFT mag. [2:] of 3D coordinates concat.	2,118,242	**0.8403**	**+0.3%**
11	RFFT mag. [1:32] of 3D coordinates concat.	2,118,242	**0.8401**	**+0.3%**
12	RFFT mag. [:31] of 3D coordinates concat.	2,118,242	**0.8374**	**$\pm$0%**
BL	-	*2,116,642*	*0.8394*	-
13	RFFT mag. [1:27] of 3D coordinates concat.	2,116,642	**0.8446**	**+0.5%**
14	RFFT phase [1:27] of 3D coordinates concat.	2,116,642	**0.8405**	**+0.1%**

Table 5.5: Re-fined frequency based methods in the CTR-GCN baseline model evaluated on NTU-120 XSub, fourier transform of embedding dimension. Modification of first TCN unit.

#	Modification	Window size [overlap]	Parameters	Accuracy	Change (%)
BL	-	*3*	*2,109,602*	*0.8398*	-
1	RFFT mag. [1:] of 3D coordinates concat., time window	8 [0]	2,109,602	**0.8403**	**+0.1%**
2	RFFT mag. [1:] of 3D coordinates concat., time window (x zero padded)	8 [1]	2,109,602	**0.8399**	**±0%**
BL	-	*3*	*2,108,962*	*0.8383*	-
3	RFFT mag. [1:] of 3D coordinates concat., time window	4 [0]	2,108,962	**0.8418**	**+0.4%**
BL	-	*64 [0]*	*2,118,562*	*0.8396*	-
4	RFFT mag. [1:] of 3D coordinates concat., embedding window	8 [0]	2,118,562	**0.8418**	**+0.4%**
5	RFFT mag. [1:] of 3D coordinates concat., embedding window	4 [0]	2,118,562	**0.8404**	**+0.3%**
BL	-	*64 [0]*	*2,119,842*	*0.8381*	-
6	RFFT mag. [1:] of 3D coordinates concat., embedding window	8[1]	2,119,842	**0.8420**	**+0.4%**

Table 5.6: Frequency based methods in the CTR-GCN baseline model evaluated on NTU-120 XSub, Window of fourier transform on embedding dim. Modification of first TCN unit.

#	Modification	Param.	Accuracy	Change (%)
BL	-	2,108,322	*0.8375*	-
1	FFT mag. [:] of 3D joint coordinates instead of input	2,108,322	0.7480	-9.0%
BL	*Bone*	*2,108,322*	*0.8497*	-
2	FFT mag. [:] of 3D bone coordinates instead of input	2,108,322	0.7493	-10.0%
BL	*Velocity*	*2,108,322*	*0.8008*	-
3	FFT mag. [:] of 3D velocity coordinates instead of input	2,108,322	0.7109	-9.0%
BL	*Double number of input channels*	*2,128,802*	*0.8378*	-
4	FFT mag. & phase [:] of 3D coordinates concat. instead of input	2,128,802	0.7431	-9.5%
BL	*Kernel size 1x1*	*2,091,938*	*0.8394*	-
5	FFT mag. [:] of 3D coordinates instead of input & kernel size 1x1	2,091,938	0.7520	-8.7%
BL	-	*2,112,482*	*0.8414*	-
6	FFT mag. [:] of 3D coordinates instead of input, FNN layer before conv	2,112,482	0.7441	-9.7%
BL	*Kernel size 3x1*	*2,104,290*	*0.8403*	-
7	FFT mag. [:] of 3D coordinates instead of input, FNN layer before conv & kernel size 3x1	2,104,290	0.7550	-8.5%
BL	*Kernel size 2x1*	*2,100,194*	*0.8417*	-
8	FFT mag. [:] of 3D coordinates instead of input, FNN layer before conv & kernel size 2x1	2,100,194	0.7509	-9.1%
BL	*Kernel size 1x1*	*2,096,098*	*0.8395*	-
9	FFT mag. [:] of 3D coordinates instead of input, FNN layer before conv & kernel size 1x1	2,096,098	0.7532	-8.6%

Table 5.7: Frequency based methods in CTR-GCN baseline model evaluated on NTU-120 XSub, fourier transform on time dimension as sole input. Modification of first TCN unit.

#	Modification	Layer [unit]	Embed.	Param.	Accuracy	Change (%)
BL	-	-	Single	2,108,322	*0.8375*	-
2	FFT mag. [:] of 3D coordinates concat. @ time dim	1 [T]	Single	2,108,322	0.8289	-0.9%
3	FFT mag. [1:] of 3D coordinates concat. @ time dim	1 [T]	Single	2,108,322	0.8348	-0.3%
4	RFFT mag. [1:] of 3D coordinates concat. @ time dim	1 [T]	Single	2,108,322	0.8322	-0.5%
5	RFFT mag. [1:] of 3D coordinates concat. @ time dim, norm by 1. coefficient	1 [T]	Single	2,108,322	0.8288	-0.9%
BL	*Double number of input channels*	*1 [T]*	*Single*	*2,128,802*	*0.8378*	-
1	FFT mag. [:] of 3D coordinates concat. @ embedding dim	1 [T]	Single	2,128,802	0.8246	-1.3%

Table 5.8: Inclusion of frequency based methods into the CTR-GCN baseline model evaluated on NTU-120 XSub, fourier transform on time dimension

#	Modification	Layer [unit]	Embed.	Param.	Accuracy	Change (%)
*	-	-	Single	2,108,322	0.8375	-0.3%
*	w/o conv initialization	1[T]	Separate	2,112,610	0.8414	+0.1%
BL	-	1[T]	Separate	2,112,610	*0.8406*	-
1	FFT mag. [:] of 3D coordinates concat. @ time dim. & conv with 1x1 kernel	1[T]	Separate	2,112,610	0.8341	-0.3%
2	FFT phase [:] of 3D coordinates concat. @ time dim. & conv with 1x1 kernel	1[T]	Separate	2,112,610	0.8396	-0.1%
3	FFT mag. [:] of 3D velocity concat. @ time dim. & conv with 1x1 kernel	1[T]	Separate	2,112,610	0.8291	-1.1%
4	FFT phase [:] of 3D velocity concat. @ time dim. & conv with 1x1 kernel	1[T]	Separate	2,112,610	0.8331	-0.8%
5	FFT mag. [:] of 3D coordinates concat. @ time dim. & conv with 1x1 kernel, no residual unit	1[T, R]	Separate	2,112,610	0.8341	-0.6%
6	FFT mag. [1:] of 3D coordinates concat. @ time dim. & conv with 1x1 kernel	1[T]	Separate	2,112,610	0.8362	-0.4%
7	RFFT mag. [1:] of 3D coordinates concat. @ time dim. & conv with 1x1 kernel	1[T]	Separate	2,112,610	0.8328	-0.8%
8	FFT mag. [:] of 3D coordinates concat. @ time dim. & conv with 1x1 kernel, Mult. in GCN	1[T], 2[G]	Separate	2,112,610	0.8389	-0.2%
9	FFT mag. [:] of 3D coordinates concat. @ time dim. & conv with 1x1 kernel, Add. in GCN	1[T], 2[G]	Separate	2,112,610	0.8389	-0.2%
10	FFT mag. [:] of 3D coordinates concat. @ time dim. & conv with 1x1 kernel, GCN with separate embedding and dim reduction by factor 2 then concat.	1[T], 2[G]	Separate	2,112,610	**0.8402**	**±0%**

#	Modification	Input	Param.	Accuracy	Change (%)
BL	-	*After 1. layer*	*2,199,986*	*0.8214*	-
1	FFT phase [:] of 3D coordinates with 2 embed. layers prior to FFT	After 1. layer	2,199,986	0.8240	**+0.3%**
BL	-	*After 1. layer*	*2,195,826*	*0.8222*	-
2	FFT phase [:] of 3D coordinates with 1 embed. layer prior to FFT	After 1. layer	2,195,826	0.8208	-0.1%
BL	-	*After 1. layer*	*2,200,114*	*0.8212*	-
3	FFT phase [:] of 3D coordinates with embed. layers before and after FFT	After 1. layer	2,200,114	0.8249	**+0.4%**
BL	-	*After 1. GCN*	*2,200,114*	*0.8269*	-
4	FFT phase [:] of 3D coordinates with embed. layers before and after FFT	After 1. GCN	2,200,114	0.8255	-0.1%
5	FFT phase [:] of 3D coordinates with embed. layers before and after FFT	After 1. TCN	2,200,114	0.8253	-0.1%
BL	-	*After 1. layer*	*2,204,402*	*0.8278*	-
8	FFT phase [:] of 3D coordinates with embed. layers before and after FFT, time reduction w/ embedding layer	After 1. layer	2,204,402	0.8254	-0.2%
BL	-	*After 1. layer*	*2,204,274*	*0.8291*	
9	FFT phase [:] of 3D coordinates with embed. layers before and after FFT, FNN for time dim reduction	After 1. layer	2,204,274	0.8173	-1.2%
BL	-	*After 1. layer*	*2,200,498*	*0.8243*	-
10	FFT phase [:] of 3D coordinates with embed. layers before and after FFT, embed. for time dim reduction. Add to include in the model	After 1. layer	2,200,498	0.8253	**+0.1%**

Table 5.10: Symmetry module experiments using CTR-GCN baseline model evaluated on NTU-120 XSub. Input at beginning of neural net and concat to result of last TCN-GCN unit as additional channel, then embedding. Fourier transform on

#	Additional input	# Parameter	Accuracy
BL	-	691,348	*87.263*
1	Sum of velocities over joint	691,348	64.800
2	Cumulative sum of velocities over joints	691,348	66.851
3	Cumulative sum of velocities over joints w/ own embedding	695,914	**88.799**
4	Cumulative sum of velocities over time	691,348	43.441
5	Cumulative sum of velocities over time w/ own embedding	695,914	**88.617**

Table 5.11: Experimental results of additional input of SGN model evaluated on NTU-60 CS

#	Input	Implementation	# Parameter	Accuracy (Epoch)
BL	*3D coordinates*	-	*2,108,322*	*83,75% (64)*
1	Cent. skeleton (3D)	-	2,108,322	tbd (?)
2	Rot., cent. skeleton (3D)	-	2,108,322	83,94 (61)
3	Spher. coordinates	cosine sim. (θ, ϕ), norm (p)	2,108,322	72.28 (57)
4	Spher. coordinates	radians (θ, ϕ), norm (p)	2,108,322	81.36 (65)
5	Cylin. coordinates	cosine sim. (θ), norm (r), value (z)	2,108,322	72.50 (57)
6	Cylin. coordinates	radian (θ), norm (r), value (z)	2,108,322	81.97 (61)
7	Azimuth, BG	cosine sim.	2,107,610	**73.45%** (38)
8	Azimuth, cent. BG	cosine sim.	2,107,610	**73.64%** (38)
8	Azimuth, cent. BG	radiams	2,107,610	**73.64%** (38)
9	Azimuth, cent.	radians	2,107,610	**75.11%** (58)
10	Colatitude	cosine sim.	2,107,610	71.61% (38)
11	Colatitude, cent.	cosine sim.	2,107,610	71.57% (37)
13	Colatitude, cent.	radians	2,107,610	tbd% (?)
14	Radius	vector norm	2,107,610	**76.03%** (38)
15	Radius, cent.	vector norm	2,107,610	**76.16%** (37)
16	Azimuth & Longitude	cosine sim.	2,107,966	**76.30%** (38)
17	Azimuth & Longitude, cent.	cosine sim.	2,107,966	**76.44%** (38)
18	Azimuth & Longitude, norm.	cosine sim.	2,107,966	74.51% (38)

Table 5.12: CSUB, default config

#	Input	Setting	# Parameter	Accuracy (Epoch)
BL	*3D coordinates*	*cset*	*2,108,322*	*85.21% (64)*
1	Cent. skeleton (3D)	-	2,108,322	tbd (?)
2	Rot., cent. skeleton (3D)	-	2,108,322	85.19% (61)
3	Spherical coordinates, own	cset	2,108,322	73.86% (59)
4	Spherical coordinates, global	cset	2,108,322	82.91% (65)
5	Cylindrical coordinates	cset	2,108,322	74.31% (58)
6	Cylindrical coordinates	cset	2,108,322	83.74% (63)
7	Azimuth, cent.	cosine sim.	2,107,610	**74.76%** (38)
8	Azimuth, cent.	radians	2,107,610	**75.57%** (62)
9	Colatitude, cent.	cosine sim.	2,107,610	72.95% (37)
10	Colatitude, cent.	radians	2,107,610	tbd% (?)
11	Radius, cent.	cset	2,107,610	**77.41%** (40)
12	Radius	cset	2,107,610	**77.32%** (39)
9	Azimuth & Longitude, cent.	cset	2,107,966	**77.76%** (61)

Table 5.13: CSET default

#	Input	Setting	# Parameter	l	Accuracy (Epoch)
BL	*3D coordinates*	-	*2,108,322*	-	*83,75% (64)*
1	Spherical coord. w/o SHT	Local	2,133,954	-	83.97% (39)
2	Spherical coord. w/ SHT	Local	2,133,954	1	80.39% (65)
3	Spherical coord. w/ SHT	Local	2,151,754	2	tbd% (?)
4	Spherical coord. w/ SHT	Local	2,178,454	1,2	83.97% (?)
4	Spherical coord. w/ SHT, non neg. only	Local	2,160,654	1,2	tbd% (?)

Table 5.14: Spherical Harmonics (CSUB, default config)

Chapter 6

Discussion and Conclusion

6.1 Summary

6.2 Future Work

Bibliography

[1] Yuxin Chen, Ziqi Zhang, Chunfeng Yuan, Bing Li, Ying Deng, and Weiming Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13359–13368, 2021.

[2] Ian R Cole. *Modelling CPV.* PhD thesis, Loughborough University, 2015.

[3] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1110–1118, 2015.

[4] EUROPEAN PARLIAMENT AND OF THE COUNCIL. Regulation (eu) 2019/2144 of the european parliament and of the council, 2019. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32019R2144.

[5] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. A new representation of skeleton sequences for 3d action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3288–3297, 2017.

[6] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C Kot. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2684–2701, 2019.

[7] Mengyuan Liu, Hong Liu, and Chen Chen. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition*, 68:346–362, 2017.

[8] Kei-ichiro Minami, Hiroshi Nakajima, and Takeshi Toyoshima. Real-time discrimination of ventricular tachyarrhythmia with fourier-transform neural network. *IEEE transactions on Biomedical Engineering*, 46(2):179–185, 1999.

[9] Mikel D Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[10] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *CVPR*, 2019.

[11] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Adasgn: Adapting joint number and model size for efficient skeleton-based action recognition. In *ICCV*, 2021.

[12] Chenyang Si, Ya Jing, Wei Wang, Liang Wang, and Tieniu Tan. Skeleton-based action recognition with spatial reasoning and temporal stack learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 103–118, 2018.

[13] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1290–1297. IEEE, 2012.

[14] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Computer vision and image understanding*, 104(2-3):249–257, 2006.

[15] Junwu Weng, Mengyuan Liu, Xudong Jiang, and Junsong Yuan. Deformable pose traversal convolution for 3d action and gesture recognition. In *Proceedings of the European conference on computer vision (ECCV)*, pages 136–152, 2018.

[16] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[17] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *Proceedings of the IEEE international conference on computer vision*, pages 2117–2126, 2017.

[18] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive neural networks for high performance skeleton-based human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1963–1978, 2019.

[19] Pengfei Zhang, Cuiling Lan, Wenjun Zeng, Junliang Xing, Jianru Xue, and Nanning Zheng. Semantics-guided neural networks for efficient skeleton-based human action recognition. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1112–1121, 2020.

[20] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Appendix A

Program Code / Resources

The source code, a documentation, some usage examples, and additional test results are available at ...

They as well as a PDF version of this thesis is also contained on the CD-ROM attached to this thesis.

Appendix B

Further Experimental Results

In the following further experimental results are ...

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Master-/Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, den 14.11.2022 Unterschrift