Universiteit Leiden

Master Computer Science

FESD: A Fault Estimation Pipeline for Human Pose Estimation

Name:	Leonardo Benedikt Pohl
Student ID:	s3059030
Date:	23/06/2023
Specialisation:	Advanced Computing and Systems
1st supervisor:	Dr. Erwin M. Bakker
2nd supervisor:	Prof. Dr. Michael S. Lew

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Human pose estimation(HPE), or skeleton detection, has been a topic of research for many years. With the advancement of hardware, it has become viable to apply HPE in real-time applications such as games. However, HPE is a difficult task that is prone to errors, especially in complicated situations, such as cramped environments. These errors might cause human-computer interaction to be hampered.

The goal of this thesis is to obtain a better understanding of what causes the problems in HPE. Therefore, various exercises and scenarios are designed that exhibit common errors in HPE. The exercises are captured in a labelled dataset. Finally, preliminary models are developed and trained on the new dataset to automatically detect the HPE errors.

The dataset is captured using a newly developed tool that is capable of capturing and labelling multi-modal data, the **F**ault **E**stimator for **S**keleton **D**etection **Data** (FESDData) tool. Four different datasets are captured using four problem sets with varying granularity, ranging from body-wise fault estimation to joint-wise fault estimation. Using the captured dataset, the FESDDataset, preliminary experiments were conducted using two new deep neural networks (DNNs), **F**ault **E**stimator for **S**keleton **D**etection **Model** V1 (FESDModelv1) and FESDModelv2. Where FESDModelv1 uses custom feature extraction, and FESDModelv2 uses transfer learning using EfficientNetv2.

Preliminary experimental results showed promising performance of both FESDModelv1 and FESDModelv2 on coarse grain scenarios such as Half Body HPE error detection.

Acknowledgement

I would like to thank my supervisor Dr. Erwin Bakker, who guided me throughout this thesis and allowed me to use a depth camera for the thesis. I would like to thank SilverFit for giving me the necessary insight into common errors in HPE, and for allowing me to use their hardware for the collection of the dataset. Additionally, I would like to thank my friends and family who have been patient enough to listen to me talk about my thesis for the past 14 months. And I would especially like to thank my partner Maria Xiberras, who has supported me throughout this thesis, even if that meant many restless weekends and delaying our plans for longer than expected.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human pose estimation (HPE), or skeleton detection, aims at detecting the pose or skeleton of a person based on visual information and/or depth information only. It finds many applications, from games to medical applications[1, 2, 3].

Gaming and entertainment are one of the most common applications of HPE. Games can use HPE in a way that makes the interaction between humans and computers very natural. One of the systems that kickstarted the use of depth cameras in games was the Xbox Kinect by Microsoft, which uses a depth camera, the Microsoft Kinect, to track the movements of the player and used this information to control the games.

HPE also finds application in *Autonomous driving*, which has been in development ever since humans replaced horses with cars[4]. However, the development of autonomous driving has been challenging. The main reason for this is that autonomous driving requires a lot of information about the environment, e.g. the road, pedestrians, other vehicles, etc. In some cases, cars need to be able to estimate the pose of a human to make a decision. The posture of a human can be used to determine the action and therefore the future trajectory of the person.

To exactly emulate human movements in *Animations*, animators can either manually move the joints of a digital skeleton or they can use HPE of real human actors. The manual creation of realistic movement is oftentimes very time-consuming and also error-prone. Therefore, animators often use HPE of real human actors leading to digital skeletons and movements that often are more accurate.

HPE is also used in medical applications, to aid in the *Rehabilitation* of patients as well as to improve the quality of life of elderly people. Here HPE can be used to detect the pose of a person and provide feedback on the pose of the person during exercises[2].

Additionally, HPE can be used as a *Diagnostic tool* for all ages. For example, to detect Cerebral palsy (CP) usually, an MRI is used using human observations. However, MRIs are expensive and the expert knowledge to detect CP requires long training. HPE offers a low-cost alternative for the detection of CP risk using automatic movement assessment with comparable performance to standardised CP risk measurers[5].

Throughout these applications, HPE is a critical component that is required to be accurate. However, HPE is a difficult task that is prone to errors[6], which can be caused by different factors, such as the environment, the camera, and the person. These errors can cause the joints of the estimated pose to be in incorrect positions or missing, thereby hampering the human-computer interaction. The goal of this thesis is to develop a method that allows for the detection of these errors such that, the pose information can be improved

by post-processing.

One of the main challenges for HPE is the diversity of users and posture appearances. If the posture is very different from what the pose estimation model expects then errors might occur. This is especially true for applications that are designed for rehabilitation and exercise purposes. In these applications, the user is often elderly and has limited mobility. This is precisely the case for games developed by SilverFit[1]. SilverFit is a serious gaming company that develops games for rehabilitation with a special focus on geriatric patients. In their games, SilverFit uses HPE to detect the pose of the player and use it to control the game to make exercising more enjoyable while promoting activity. The research conducted in this thesis is inspired by the challenges that the applications from SilverFit have to face and aims at improving HPE by giving feedback on the detection of errors occurring during HPE in their games.

For some exercises designed by SilverFit and other companies, HPE is not sufficiently reliable to create an enjoyable experience for patients in every environment. Especially sedentary exercises often cause the HPE to fail or to produce unreliable results.

To analyse and mitigate the errors that occur during HPE three research questions are formulated.

(A) What typical errors occur during HPE and how can these errors be reproduced in controlled scenarios?

(B) How can a multimodal HPE dataset be captured and labelled?

(C) Can we train a deep neural network (DNN) to detect errors in human pose estimations using RGB-D and pose input data?

To answer the research questions, first, the most common HPE error sources are analysed. During this analysis, different factors are discussed which may influence the performance of human pose estimation. Based on these factors exercises are designed to emulate different scenarios with different levels of challenges for HPE. This is discussed in Section 4.1.

In Section 4.2 a custom tool, FESDData, was designed and implementeted to capture and label multi-modal data, consisting of RGB, Depth and Pose data, and metadata such as exercise name and frame time stamps. The tool allows capturing predefined exercises and labelling each joint of each captured pose data frame with error labels. The collected data forms the dataset, FESDDataset.

In Section 5, two DNN models for HPE error detection are proposed, FESDModelv1 and FESDModelv2. The FESDDataset is used as the training set for these models using a data loader that performs the necessary preprocessing steps, such as data augmentation, data balancing and resizing of the modalities. FESDModelv1 and FESDModelv2, are trained to detect HPE errors at different levels of difficulty. The experimental setup is described in Section 5.3. Finally, the experimental results are discussed in Section 6.

[1] https://www.silverfit.com/en/

Chapter 2

Related Work

Human Pose Estimation (HPE) Multiple different human pose estimators have been developed using different modalities and focussing on different parts of the body.

OpenPose has developed a general human pose estimator[7], a hand pose estimator[8] and an estimator that is capable of detecting multiple people[9]. This is achieved by using RGB data only.

A method that uses RGB data in combination with errors for estimating the correct pose, is proposed by Joao Carreira et al.[10]. In their paper, the authors extract the features of both input and output spaces. This enables them to develop a self-correcting model using Iterative Error Feedback. To predict the correct pose for an image, an initial guess is made. Based on this guess, the most optimal predefined error correction is applied and the pose is updated iteratively until a threshold is reached. This method relies on the accuracy of an evaluator or a predefined number of iterations to determine a pose. The method might benefit from a dedicated error detector as is proposed in this research.

Other methods use point clouds which are extracted from depth data. Point clouds are a representation of depth in the environment. These are commonly used in autonomous driving to represent the real world more accurately and to describe complex objects in the environment. However, conventional feature extraction using CNNs proves difficult to be applied on point clouds since most point clouds are not ordered and not dense so locality cannot be ensured. PointCNN, proposed by Yangyan Li et al.[11], combats this problem by learning an $\mathcal{X}$-Transformation to then extract the features using a generalised CNN.

One of the uses of HPE is action recognition. Action recognition is the detection of predefined actions from different modalities. Seddik et al. compare different fusion methods for action recognition[12]. After detecting the features for each of the modalities, RGB data, Depth data and Pose data, they fuse the features using different methods.

In this thesis, we use RGB data, and depth data in addition to pose data to detect the errors created by human pose estimation. We are unaware of any previous work specifically focused on HPE error detection using DNNs.

Datasets There are different datasets for human pose estimation encompassing different environments and modalities. The KITTI dataset is captured using LIDaR cameras and contains both depth and RGB data[13]. One of the largest datasets specifically for HPE and its applications is Human3.6M[14]. Human3.6M is a large-scale dataset containing RGB, Depth and Pose information.

To our knowledge, there does not exist a dataset that specifically focuses on error de-

tection in human pose estimation.

Chapter 3

Fundamentals

For a better understanding of error detection in HPE, several fundamentals in HPE are discussed in this section.

3.1 Human pose estimation

There is a wide variety of how humans can interact with computers. Ranging from early punch cards to touch screens, the methods have evolved to be more natural and intuitive. In recent years, the use of cameras has become more popular, since they require no physical contact with a computer and therefore allow users to seamlessly interact with an application.

Pose visualisation Different methods to estimate the pose of a human have been developed, such as OpenPose[7], or Nuitrack[1]. Depending on the usage of the pose estimation, different methods are used to visualise the pose of a human. These visualisations may provide different information about the pose of a human.

There are mainly three different ways the human pose can be visualised. The first and most basic way is to visualise the pose as a kinematic representation, in which a "skeleton" with bones and joints is used to represent the pose of a human. The skeleton is made up of joints, which are connected by bones. The number of joints and bones can vary, for example, the representation shown in Figure 3-1(a) uses 15 joints and 14 bones. The representation of a joint in the data varies, but it is usually a 2D point on the image plane or a 3D point in space, for example in a point cloud. In some cases, the joint representation is provided with a keypoint orientation that enables the clear representation of all degrees of freedoms joints have[15]. Additionally, in some cases, especially if the human pose was estimated using a neural network, a confidence rating or score is added which can be used to determine the reliability of the joint. In this work, the confidence rating is not used as it is not generally applicable and unreliable in the method of pose estimation that was chosen.

The second way to visualise a human pose is by using a 2D silhouette or 2D rectangles and shapes. These methods are also called contour-based methods. An example of contour-based methods was introduced by Yunheng Liu[16]. Contour-based methods are often used in combination with a skeleton representation. The skeleton is used to determine the location of the joints, while the contour is used to determine the shape of the body. The silhouette representation is visualised in Figure 3-1(b).

[1] https://nuitrack.com/

Finally, the third way to represent a human pose is with a three-dimensional volume this is shown in Figure 3-1(c). This volume may be simple cylindrical shapes or a body mesh. A body mesh is a 3D representation of the body, which is made up of vertices and triangles.

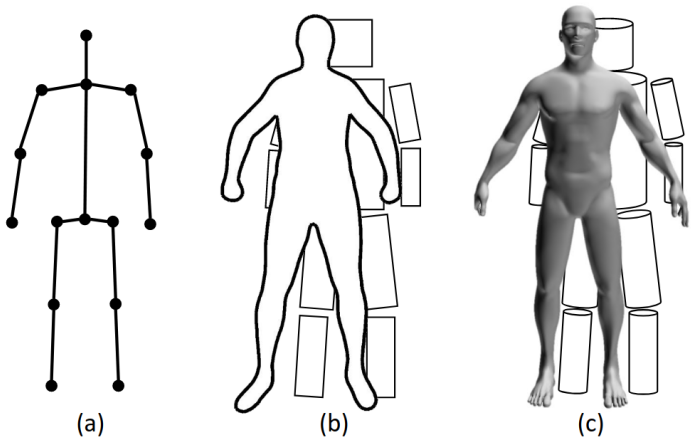


Figure 3-1: Different representations of the human pose. (a) Kinematic representation. (b) Contour representation. (c) 3D volume representation. [17]

In this work, the kinematic representation of a pose was chosen, as it provides the most efficient way of capturing pose information while retaining the most essential information about the pose.

Data sources The data that is used as an input for HPE influences the way the human pose can be estimated. The most common data source consists of RGB images or videos. RGB videos are captured with normal cameras that record the colour of the scene. The provided data can be either from a video or a stream of data or a still image. There is a large number of datasets that can be used to train and test poses estimation algorithms from RGB data. Some of the most common datasets are the MPII Human Pose Dataset[18], the COCO dataset[19], and HumanEva-I dataset[20].

Additionally, some datasets are captured with depth cameras. Depth cameras are cameras that can record, in addition to the colour channels, the depth of the scene. Different methods to acquire depth data have been developed. The most commonly used depth cameras emit an infrared light pattern to determine the depth of a scene. This depth information can be used to improve the accuracy of the pose estimation. Some of the most common datasets that are captured with depth cameras are the MRI dataset[21], and the Human3.6M dataset[14].

Finally, there are also methods of HPE that use point clouds. Point clouds are a collection of points in space, which can be used to represent the shape of an object. Each point in the point cloud represents a point in the environment. The point cloud is created using the depth information of a scene and the intrinsics of a camera, i.e. the field of view and the focal point of the camera. A point cloud is a reconstruction of the real-world scene. Point clouds are often used in combination with RGB data. Some of the most common datasets that are captured with depth cameras are the SMMC-10 dataset[22], and the EVAL dataset[23]. However, any RGB and depth dataset can be used to train and test point cloud-based pose estimation algorithms if the camera intrinsics and extrinsic, such as

the horizontal and vertical field of view, the focal point, and the depth units are known. With the knowledge of these parameters, the depth information can be converted to a point cloud and if the RGB data is in line with the depth data, the individual points can be coloured accordingly.

However, these datasets are not aimed for error detection and it cannot be ensured that they are suitable for error detection.

3.2 Convolutional Neural Networks

Convolutional neural networks(CNN) are a type of feed-forward neural network that is commonly used for computer vision tasks. As the name suggests, CNNs apply convolutional operations in some of the layers of the network architecture. A convolutional operation is an integral over two functions to determine the amount of overlap of these functions, resulting in a third function. In CNN, the first function is usually an image or rather a matrix representation of an image and the second function is a matrix, that is changed throughout the course of network training. This operation then results in a new matrix. This is used to extract local features from an image.

In most cases, the features that are produced by multiple convolutional layers are forwarded into fully connected layers. These layers learn the features based on the input data. In the case of image classification tasks, the fully connected layers are usually followed by a softmax layer to calculate the probability of each class. The class with the highest probability is then used as the prediction of the network for the given input.

Since convolutional neural networks require a lot of data to be trained properly, transfer learning is applied in some cases. Transfer learning is a technique that uses a pre-trained model to extract features from the input data. The pre-trained model is usually trained on a large dataset, such as ImageNet[24]. The pre-trained model is then used to extract features from the input data. These features are then used to train a new model. This allows the new model to be trained with less data and therefore less time. Additionally, the pre-trained model is usually trained on a large dataset and therefore the features that are extracted are more general and can be used for different tasks.

3.3 Evaluation metrics

To evaluate the performance and accuracy of a model, different metrics are used. In this section, some metrics that are used to evaluate the performance of our models are discussed.

3.3.1 Confusion Matrix

For binary classification problems, a confusion matrix consists of the numbers of the true/-false positives and the true/false negatives in a single matrix. The confusion matrix can be seen in Table 3.1. The confusion matrix is useful to visualise the performance of a model.

3.3.2 Percentage of positive Guesses

The percentage of positive guesses is an indicator of the performance of a model. For example, if the dataset is unbalanced, the accuracy might be sufficiently good when only guessing for the majority class. In those cases, the percentage of positive guesses would either be 0% or 100%. Therefore, the percentage of positive guesses is used to determine

Table 3.1: The confusion matrix as a metric for binary classification. The results of the prediction (PRED), left, and the actual ground truth (GT), top.

PRED \GT	TRUE	FALSE
TRUE	TP	FP
FALSE	FN	TN

if the model is biased toward a specific class. The equation for the percentage of positive guesses can be seen in Equation 3.1. In this thesis, the percentage of positive guesses is sometimes referred to as $\frac{p}{p+n}$ as it is the ratio of the number of positive guesses p to the total number of guesses $p+n$, where n is the number of negative guesses.

$$\text{Percentage of positive guesses} = \frac{\text{Number of positive guesses}}{\text{Number of guesses}} = \frac{tp+fp}{tp+fp+tn+fn} \tag{3.1}$$

3.3.3 Receiver Operating Characteristic Curve

The receiver operating characteristic curve (ROC curve) is a metric that shows the performance of a model performing binary classification tasks. It plots the true positive rate $\frac{TP}{TP+FN}$ against the false positive rate $\frac{FP}{FP+TN}$. This allows for the evaluation of a classifier without the influence of class distribution. An interpretation of the ROC Curve can be seen in Figure 3-2

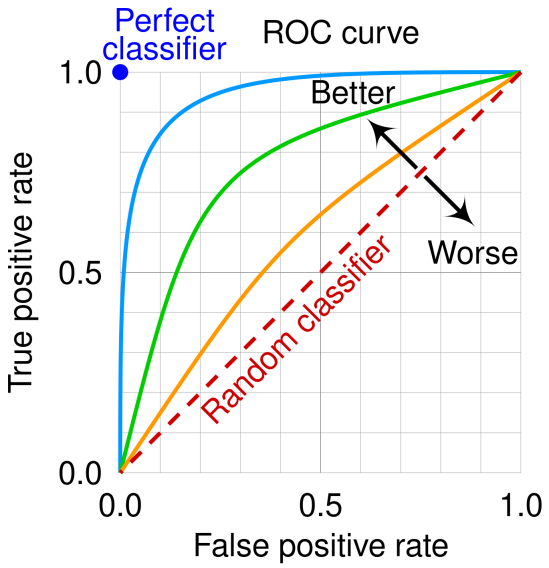


Figure 3-2: An interpretation of the ROC curve. The plotted lines are examples of exceedingly good classifiers, with orange being the worst and blue being the best.[25]

3.3.4 Accuracy, Precision, Recall and F1-Score

The accuracy of a prediction is the ratio of the number of correct predictions to the total number of predictions. The accuracy is calculated using Equation 3.2. The accuracy is a

good indicator of the performance of a model if the dataset is balanced. However, if the dataset is unbalanced, the accuracy might be sufficiently good when only guessing for the majority class.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Number of predictions}} = \frac{tp + tn}{tp + fp + tn + fn} \tag{3.2}$$

The precision of a prediction is the ratio of the number of correct positive predictions to the total number of positive predictions. The precision is calculated using Equation 3.3.

$$\text{Precision} = \frac{\text{Number of correct positive predictions}}{\text{Number of positive predictions}} = \frac{tp}{tp + fp} \tag{3.3}$$

The recall of a prediction is the ratio of the number of correct positive predictions to the total number of positive samples. The recall is calculated using Equation 3.4.

$$\text{Recall} = \frac{\text{Number of correct positive predictions}}{\text{Number of positive samples}} = \frac{tp}{tp + fn} \tag{3.4}$$

Finally, the F1-Score is the harmonic mean of the precision and the recall. The F1-Score is calculated using Equation 3.5. The F1-Score is a good indicator of the performance of a model if the dataset is unbalanced since it takes both the precision and the recall into account.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot tp}{2 \cdot tp + fp + fn} \tag{3.5}$$

3.3.5 Cohen's kappa coefficient

Cohen's kappa coefficient is an inter-rater reliability measure[26]. This means that two different raters are compared. In the case of a binary classification, the formula for Cohen's Kappa can be seen in Equation 3.6. In the case of prediction, the two raters are the ground truth and the prediction.

$$\kappa = \frac{2(tp \cdot tn - fn \cdot fp)}{(tp + fp)(fp + tn) + (tp + fn)(fn + tn)} \tag{3.6}$$

If Kappa is less than 0, there is no agreement between the two raters. If the kappa score is less than 0.5 there is a slight agreement. A score of more than 0.8 is considered almost perfect. However, the "exact" values vary from application to application. This metric proved essential when investigating the reliability of results, as it shows a measure of reliability which is essential for error detection.

Chapter 4

FESDData - Data Acquisition and Labelling of FESDDataset

One of the most important aspects of machine learning is data acquisition since a machine learning algorithm can only be as good as the data provided. In this thesis, a custom tool is developed which is capable of capturing and labelling multi-modal data. The tool is called **F**ault **E**stimator for **S**keleton **D**etection **Data** processor (FESDData). It is capable of capturing RGB-D data (RGB and the corresponding depth) from multiple different RGB-D cameras as well as pose data calculated by Nuitrack, which was developed by 3DiVi Inc[1]. Nuitrack was chosen as the pose estimation backbone since it provided the support of multiple different camera models as well as the ability to extract the RGB-D data and pose data simultaneously.

In addition to Nuitrack, FESDData is capable of calculating the pose information using OpenPose after the recording is done. This offers future support for other pose estimation backbones. However, OpenPose is not used in the experiments for this thesis.

FESDData is designed to be easily used by anyone and to require minimal setup or tweaking. It allows for the rapid capture of RGB-D datasets with the pre-labelling of multiple different exercises. The parameters can be adapted to capture datasets for many different applications, e.g. action detection, where HPE is used to identify an action.

For this research, selected exercises which are designed to emulate different scenarios with different levels of difficulty. These selected exercises are discussed in Section 4.1. The labelling of errors in the data requires domain-specific knowledge and is therefore done manually. The resulting dataset, FESDDataset, is discussed in Section 4.7. It consists of the RGB-D data as well as the pose data and the labels for the errors for each of the joints in the pose.

In this chapter, the approach to the design of the exercises is discussed by showing the challenges of HPE. Furthermore, the data acquisition and labelling process is discussed. Finally, the features of the dataset are presented.

4.1 Challenges in HPE

In this chapter, possible faults and difficulties that occur during HPE are discussed. These difficulties are caused by different factors, such as the environment, the camera, the person, and the software.

[1] https://nuitrack.com/

These difficulties are important to understand, as they can cause the estimation of the joint postion to be in the incorrect position or missing. Leading to an incorrect estimation of the pose, and therefore, the human-computer interaction will be hampered.

4.1.1 Environment

The environment in which the data is recorded is crucial to the quality of the pose estimation. The mentioned difficulties depend on how the method itself works. For example, if a method uses RGB data, the background might be an issue, whereas if a method uses depth data, the reflections and IR component of the lighting is often more of an issue.

Background

The background of the scene can cause difficulties in the HPE process. Methods using RGB cameras might have difficulties detecting the difference between the foreground and the background. In RGB-D-based methods, the background can cause depth ambiguities which might even prevent the human from being detectable, especially if the background is reflective, or too close to the user.

Lighting

While RBG cameras are to some extent insensitive to the amount of incoming light, RGB-D cameras are highly influenced by the lighting. Most RGB-D cameras use infrared light to determine the depth of the scene, some use a pattern of infrared light that is projected onto the scene to be distorted by physical objects, and some use the time-of-flight method to determine the depth of the scene. The issue that arises with infrared light is that it is also emitted by the sun. This means that light emitted by the sun can interfere with the detection of the infrared light emitted by the camera. This can cause the depth of the scene to be inaccurate or missing in parts with a high intensity of sunlight.

To reduce the effect of the sunlight, the camera can be placed in a room with curtains or blinds. This will reduce the amount of sunlight that enters the room and therefore reduce the effect of the sunlight on the camera. Since lighting is hard to perfectly reproduce without a controlled environment, the lighting is not varied throughout the experiments. Therefore, all of the experiments were conducted in a room without any natural light or during the night.

However, if a method heavily relies on RGB data for the pose estimation, eliminating any form of light is not a valid option since then the data that can be gathered by RGB cameras in low light environments is limited and may result in a wrong pose estimation.

Objects

The objects in the scene may cause issues in the HPE process if they occlude or are too close to the user. Occlusion can cause inaccurate or missing joints. Whereas objects that are too close to the user can cause joints to move to these objects instead.

Chair

If the exercise is performed in a sitting position the chair might influence the accuracy of HPE. For example, wheelchairs pose a problem for some estimators, but a significant part of

users who use pose estimation for rehabilitation games are using wheelchairs due to health conditions. Furthermore, bulky chairs that go higher than the head may prevent accurate head detection and therefore, negatively impact the pose estimation, since in some pose estimators the head is used as an ankor point as it is usually reliable.

4.1.2 Camera

The two main difficulties that can occur with the camera setup are the camera position and the camera angle. Additionally, the camera itself can make the pose estimation process more difficult, as it is the means of capturing the data. If the camera does not record the data at a high enough resolution, the detection of the human pose becomes more challenging.

Distance to the camera

The distance of the camera to the user affects the quality of the pose estimation in multiple ways. Firstly, if the user is too close to the camera, the camera might not have the complete body in the frame. The legs and arms might be out of the frame preventing them to be estimated properly. Additionally, depth cameras can only detect the depth for a specific range, so if a user is too close they might not be visible to the depth camera.

However, if the user is too far away from the camera, the person might be too small to be detected reliably, or the user will not be visible to the depth camera. This range varies from camera to camera and depends on the method of detection.

Camera Angle

When considering angles the main focus lies on pitch and roll. For the experiments, the yaw is assumed as fixed and directed facing the user, such that the user is in the centre. The camera is set up such that there is no or minimal roll as all applications ensure that the camera is not tilted to the side.

The pitch may introduce or reduce the occlusion of joints by other joints. It also influences which area is best for HPE. The pitch of a camera depends on what the main application is. For example, if the legs are not considered then the pitch could be used to focus more on the upper body.

Camera Resolution

The resolution of the camera, or rather the resolution of the RGB-D image captured by the camera influences the information that can be gathered about the human and therefore influence the performance of the pose estimation. Also here the application and specific range are important for choosing the right resolution. If a user is far away a higher resolution is needed to detect the pose reliably.

4.1.3 Person

Finally, one of the main error sources of HPE is the person. The person can cause difficulties in the HPE process by moving, wearing specific clothes, or having a different body posture. Body posture is of special importance for SilverFit since the company specialises in games for rehabilitation and elderly people. Elderly people have different body postures than the average person, which can cause difficulties in the HPE process.

Clothes

As mentioned earlier, most RGB-D cameras use infrared light to determine the depth of the scene. This means that the clothes of the user can cause more or less absorption of light and therefore influence the detected depth. This can cause the joints to be detected in the wrong position or not at all. This is especially the case for dark clothes, as they absorb more light than light clothes.

Furthermore, bulky clothes or skirts and dresses may influence the pose, since the exact position of the legs is not visible.

Training Equipment

To make exercises more challenging some physiotherapists use additional training equipment. This could be weights that are held in the hand or weights that are attached to the ankles. These weights change the outline of the body and therefore influence the pose estimation.

Exercises

Finally, the most important factor is the exercise that is carried out. In this section, a range of exercises are proposed, going from easy to hard for human pose estimators to estimate.

These exercises try to cover common issues with pose estimation in a reproducible manner. The difficulty rating of the exercise does not reflect the difficulty of the exercise for the user, but it does reflect the difficulty of the exercise for the pose estimator.

The exercises are numbered according to this difficulty. An example of the naming of the exercises can be seen here:

$$\underbrace{E}_{\text{Exercise}} - \overbrace{2}^{\text{Difficulty}} . \underbrace{02}_{\text{Exercise Id}}$$

The different difficulties are (0) Trivial, (1) Easy, (2) Medium, and (3) Hard. The Exercise Id is the numbering of the exercises, not necessarily in order of difficulty.

Trivial Exercises Trivial exercises are exercises that are easy to detect for a human pose estimator and are therefore good for testing the pose estimators. The exercises do not involve any movement, which makes detecting the joints easier. The two trivial exercises can be seen in Table 4.1. Example images of the exercises can be seen in Figure 4-1.

Table 4.1: The two Trivial Exercises, E-0.00 and E-0.01.

Exercise	Short Description	Challenge
E-0.00	Arms hanging to the side	This exercise is very easy for HPE since no part of the body is occluded by any other part. Additionally, the joints are not moving.
E-0.01	Arms extended to the side	The arms are no longer in a natural position but still not occluded and no joint is moving

(a) E-0.00

(b) E-0.01

Figure 4-1: The two trivial exercises. (4-1a) Arms hanging to the side. (4-1b) Arms extended to the side.

Easy Exercises Easy exercises are essential for creating a good baseline of how the pose estimators should work. The exercises include no self-occlusion and are recorded in a standing position, which is generally the easiest position to estimate. The easy exercises can be seen in Table 4.2. Example images of the exercises can be seen in Figure 4-2.

Table 4.2: The Easy exercises, E-1.00 through E-1.03. The easy exercises include both standing and sitting exercises.

Exercise	Short Description	Challenge
E-1.00	Raising the arms to the side	Now the arms are no longer stationary. Still, there is no occlusion.
E-1.01	Raising the arms to the front	The arms occlude themselves and part of the body.
E-1.02	Raise the knees to the chest	The legs now occlude parts of the hip.
E-1.03	Sit in a chair motionless	Sitting positions are more challenging to detect than standing positions since there is now a chair in the background and the body occludes itself.

Medium Exercises Exercises which are performed in a seated position are harder for HPE since parts of the body are occluded. Medium exercises focus on exercises, which are performed in a seated position. The medium exercises can be seen in Table 4.3. Example images of the exercises can be seen in Figure 4-3.

Difficult Exercises Difficult exercises are exercises that are performed in a standing position and involve leg movement. Leg joints are harder to detect than arm joints and therefore pose a greater challenge for pose estimators. These exercises will be in a seating position and with a difficult posture, such as leaning forward. The difference in posture aims at creating a realistic representation of real-world elderly and therapeutic exercises.

Tölgyessy et al.[6] found that facing away from the camera decreases the accuracy of HPE due to self-occlusion. Therefore, some of the exercises will be performed facing away from the camera. The difficult exercises can be seen in Table 4.4. Example images of the exercises can be seen in Figure 4-4.

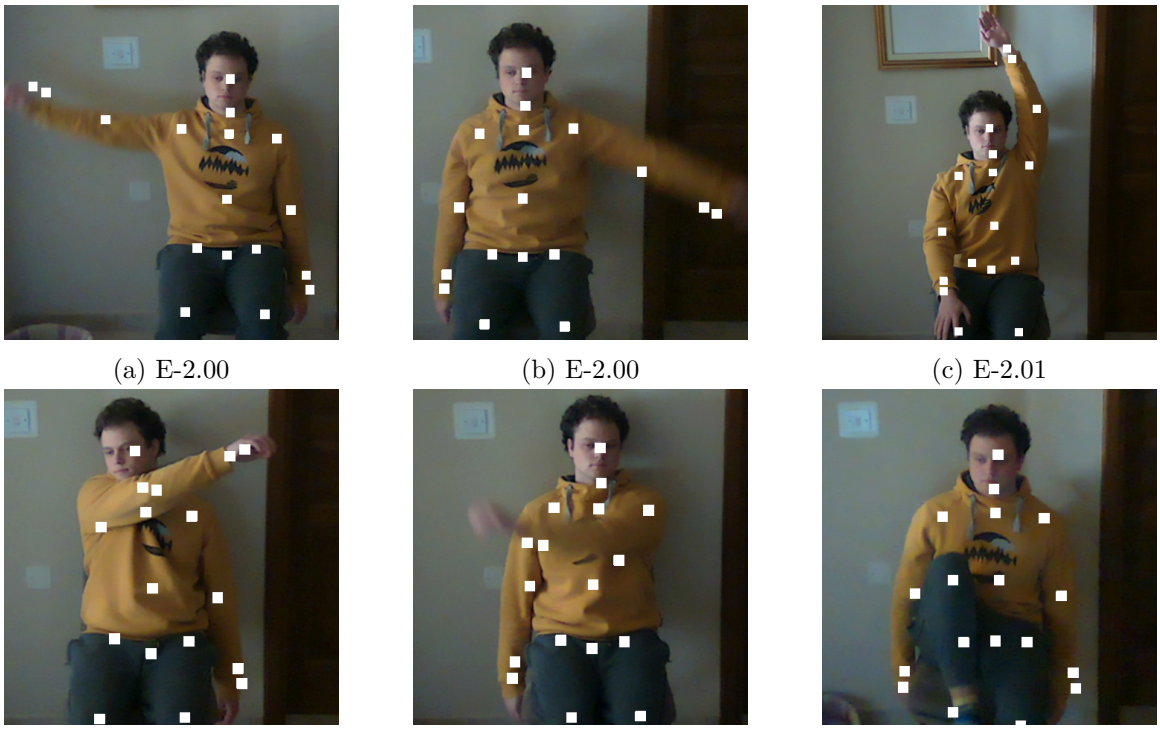
(a) E-1.00 (b) E-1.00 (c) E-1.01

(d) E-1.01 (e) E-1.02 (f) E-1.03

Figure 4-2: The four easy exercises. (4-2a, 4-2b) Raising the arms to the side. (4-2c, 4-2d) Raising the arms to the front. (4-2e) Raising the knee to the chest. (4-2f) Sitting in a chair motionless.

Table 4.3: The Medium exercises, E-2.00 through E-2.03. The medium exercises are all in a sitting position.

Exercise	**Short Description**	**Challenge**
E-2.00	Raising the arms to the side while sitting	Now the arms are no longer stationary. With added difficulty since the user is now sitting.
E-2.01	Raising the arms to the front while sitting	The arms might occlude themselves and part of the body. Additionally, the person is now sitting down
E-2.02	Crossing the arms in front of the chest while sitting	The arms now occlude large parts of the upper body.
E-2.03	Raising the knee to the chest while sitting	More parts of the upper body are occluded and not seen by the camera.



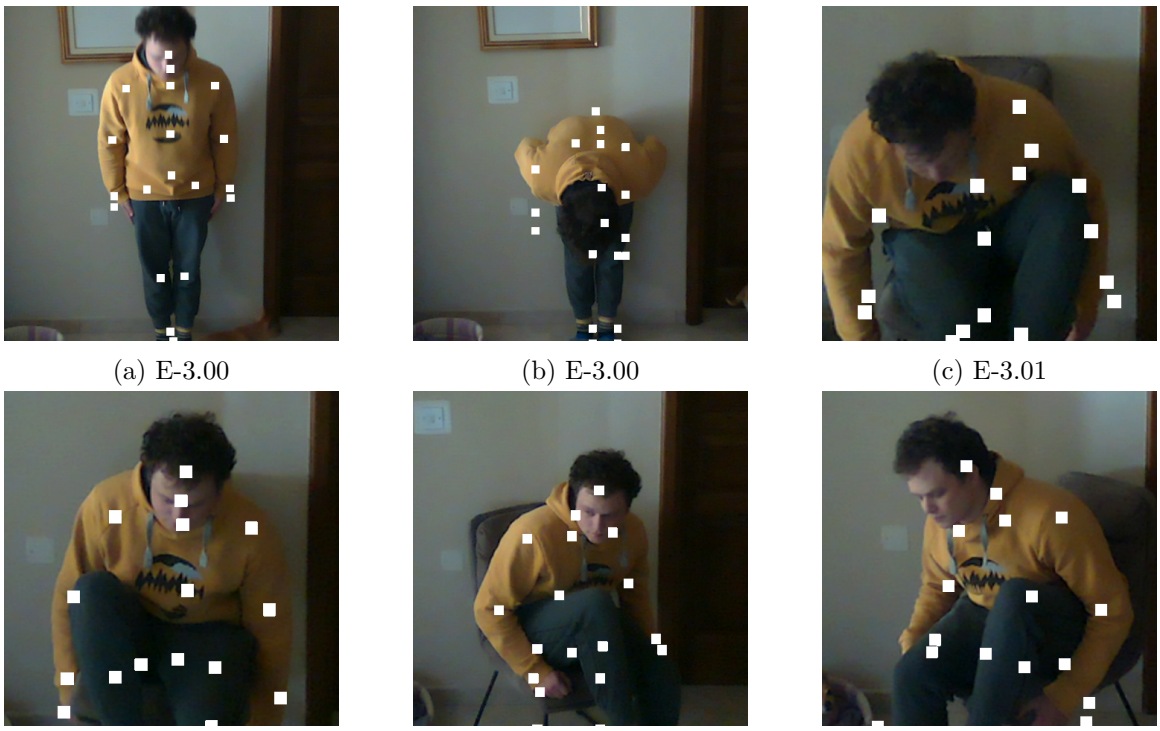
(a) E-2.00 (b) E-2.00 (c) E-2.01

(d) E-2.02 (e) E-2.02 (f) E-2.03

Figure 4-3: The four medium exercises. (4-3a, 4-3b) Raising the arms to the side while sitting. (4-3c) Raising the arms to the front while sitting. (4-3d, 4-3e) Crossing the arms in front of the chest while sitting. (4-3f) Raising the knee up to the chest while sitting.

Table 4.4: The Difficult exercises, E-3.00 through E-3.02. The difficult exercises are all in a standing position.

Exercise	**Short Description**	**Challenge**
E-3.00	Bowing toward the camera	The head is often used as an anchor point for the skeleton as it is quite stable. When bowing forward the head is no longer easily detectable.
E-3.01	Raising the knee leaning forward	Leaning forward increases the difficulty since the position is not natural
E-2.02	Raising the knee leaning forward while sitting and facing away from the camera	Facing away from the camera makes it more difficult to detect the pose and induces more occlusion.



(a) E-3.00 (b) E-3.00 (c) E-3.01

(d) E-3.01 (e) E-3.02 (f) E-3.02

Figure 4-4: The three difficult exercises. (4-4a 4-4b) Bowing toward the camera. (4-4c, 4-4d) Raising the knee leaning forward. (4-4e, 4-4f) Raising the knee leaning forward while sitting and facing away from the camera. Some of the exercises depict errors that occur during HPE.

4.2 Data acquisition

Different modalities were captured to create a dataset that reproduces a real-world application of HPE for RGB-D cameras, i.e. cameras that capture RGB and depth. The different modalities are RGB data, depth data, and joint data. While the Nuitrack SDK offers to capture the data from the RGB-D cameras and the joint data, the recorded files cannot, at the time of writing, be read without using the Nuitrack SDK. Therefore, FESDData, a custom RGB-D+HPE capturing and labelling tool was developed.

FESDData has two main uses. Firstly, it allows capturing predefined, as well as custom, exercises repeatedly automatically making the capturing experience, when capturing many different exercises with multiple repetitions, more user-friendly. Secondly, it allows reviewing and labelling the captured data with error labels. The lightweight nature of FESDData allows it to seamlessly capture both the RGB-D stream and the skeleton data at the same time while ensuring a stable fast framerate. The dataset that is used by FESDModel, which is introduced in Section 5, was captured at a framerate of 30 frames per second. The interface of FESDData can be seen in Figure 4-5 and 4-6. In Figure 4-5 the interface can be seen while streaming live or recorded data, and in Figure 4-6, an example is shown for data labelling of Exercise E1-02.

4.3 Data layout

As mentioned earlier, the dataset is made up of multiple different modalities. There are two separate visual streams, the RGB stream, and the depth stream, as well as the estimated human pose, the relative time stamps of each frame, and the recording metadata. The recording metadata includes information about the camera, such as the horizontal as well as the vertical field of view, and the exercise. The exercise is captured with a given Id and a separate file exists containing all exercises with a description.

The visual streams are normalised and combined into a single file. OpenCV is used to store the RGB and the depth data into a single matrix and after the stream into a single file per frame. The RGB data is normalised to have values between 0 and 1, whereas the depth data is stored in meters.

The pose that is estimated by Nuitrack, as well as the error labels, are stored in a separate JSON file. The separate frames are stored in a list of frames. Each frame contains a list of all people that were detected. Each person contains a list of joints as well as an error label, and an Id. Each joint is stored with real-world 3D coordinates which are stored in meters. These real-world coordinates are labelled x, y, and z, relative to the position of the camera. Additionally, the 2D projection on the image plane and the depth of the estimated joints are stored in image coordinates and meters for the depth. The coordinates of the 2D projection are labelled u and v, where u is the horizontal and v is the vertical component of the 2D projection, and the depth is labelled d.

The error label is an integer which is the error id specific for joint and skeleton errors. The errors corresponding to the error ids are further explained in Section 4.4.

4.4 Data labeling

A large part of the data preparation is the labelling of the data. The data is labelled with error labels. There are two different layers which can be labelled as erroneous. First, there

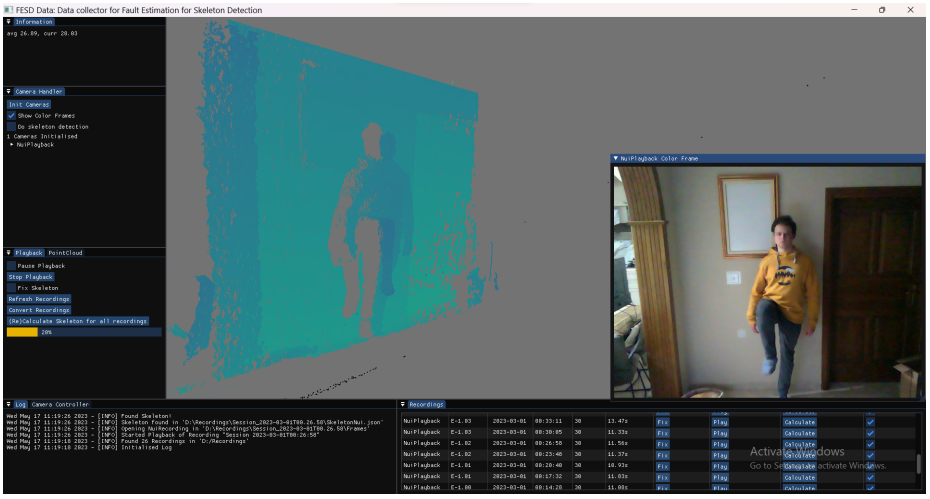


Figure 4-5: The user interface of FESDData can be seen when streaming either a recording or a live feed of a camera. Both the RGB image, as well as the projected point cloud can be seen. At the bottom of the image, the list of recordings can be seen with a flag that indicates if the data has been labelled.

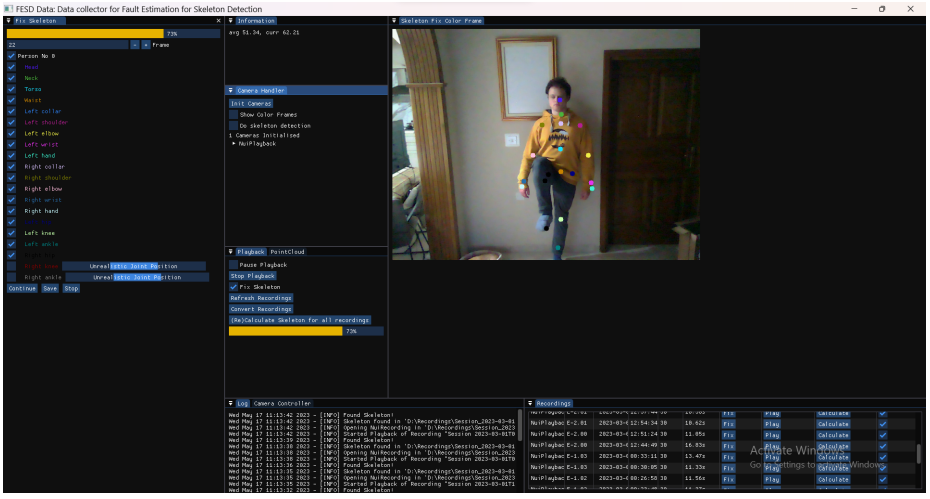


Figure 4-6: The user interface of FESDData during the labelling process. On the left, the actual labelling can be observed, where each joint, coloured according to the colour in the image, can be marked as erroneous or not. In this case, the left and right foot are in an incorrect position and are labelled as such. The right side contains a depiction of the joint positions as an overlay over the RGB image.

are skeleton errors that occur when the pose estimator detects a human in places where there are no humans. Second, there are joint errors that occur when the pose estimator detects a joint at the wrong place.

For example, the estimator might label the left foot as the right foot. This is a common error, especially when the body parts are close to each other. An estimator might also not detect a joint at all. This might be caused by occlusion, be it by another joint, an object, or by the image border. Most applications avoid the last cause for occlusion by defining a minimum distance from the camera and specific camera placement to ensure that the user is always fully in view.

The possible error labels for the skeleton itself can be seen in Table 4.5. Table 4.6, shows the different error labels for the individual joints of the skeleton.

Table 4.5: The two possible errors for the skeleton

Label	**Error Name**	**Example**
0	No Error	The skeleton is exactly aligned to the body
1	Error	The skeleton is at a different location and not on a body

Table 4.6: The four possible errors for the individual Joints.

Label	**Error Name**	**Example**
0	No Error	The joint is exactly aligned to the position where it is supposed to be
1	Missing Joint	The joint is not detected at all
2	Wrong Joint Position	The joint is somewhere outside the body
3	Different Joint Position	The joint for the left foot is at the position of the right foot

Implicitly, the errors shown in Table 4.6 create two simpler general labels. Either a joint is faulty, i.e. the error label is 1, 2, or 3, or it is not faulty, i.e. the error label is 0.

4.5 Problem Sets

When considering the problem of error detection in human pose estimation a distinction can be made, as to what is considered an error. While the data is labelled in a way that allows for the extraction of joint-level errors, such a fine-grain application is rarely necessary. In addition to this, joint-level error detection creates a very large search space. This may prove difficult when developing a model for such a task. Therefore, problem sets, which create different levels of abstraction of the area that is considered are defined.

The problem sets are defined by the number of objects that are considered and are defined as erroneous. There are four different problem sets.

(A) Joint problem set - each joint is considered as a single object

(B) Body Part problem set - each body part is considered, i.e. the individual arms, legs, torso, and head

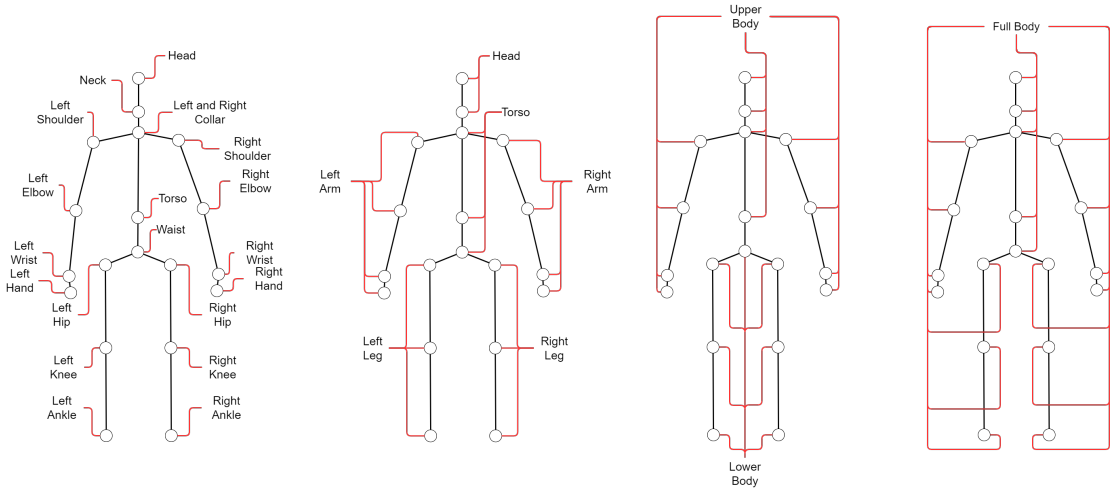
(C) Half Body problem set - the upper and the lower body are the areas that are considered

(D) Full Body problem set - the whole body is considered as a single area

To determine the threshold at which each area is considered faulty, the distribution of joints with errors was calculated for each of the areas. Using this distribution the threshold was picked at the 50th percentile, i.e. 50% of all cases for a particular area in a problem set containing more than that number of errors. For example, in the case of the Full Body problem set, the chosen threshold is 2, i.e. in 50% of all recorded frames there are two or fewer erroneous joints in the whole body.

It was found that when considering the Full Body as an area, the body is considered faulty if more than two joints in the pose are faulty. When considering the lower and upper body, one or more and more than two faulty joints are needed for the area to be considered faulty. For the body parts to be considered faulty one joint within the specific part needs to be faulty, except for the right and left leg, where two joints need to be faulty.

The different problem sets are visualised in Figure 4-7.



(a) Joint Problem Set

(b) Body Part Problem Set

(c) Half Body Problem Set

(d) Full Body Problem Set

Figure 4-7: The visualisation of the different problem sets, (a) Joint Problem Set, (b) Body Part Problem Set, (c) Half Body Problem Set, and (d) Full Bod Problem Set.

4.6 Recording Setup

Multiple iterations of the recording process were run to find the best possible setup, which reduces the light interference as much as possible and which offers the best results with the resources at hand. The camera setup that is used by SilverFit was reproduced. At SilverFit in most cases, an Orbbec Astra+ camera is mounted at 175cm above the floor. The camera that is used in the experiments has an accelerometer which was used to adjust the camera angle relative to the ground. The camera is angled downward at a 70° angle. To record the dataset an Intel Realsense L515 camera was used. Additionally, for the preliminary analysis of errors an Orbbec Astra+ and a Microsoft Kinect v2 were used.

4.7 Dataset Analysis

To get a better understanding of the data and how it can be used we conducted a statistical analysis. Especially the distribution of errors within the dataset is important to understand the balance of the dataset and to understand the possible outcomes of our proposed models for error detection in human pose estimation.

4.7.1 Distribution of Errors

An important aspect of the dataset is the structure and distribution of data and their labels. In total, all 13 exercises mentioned in Section 4.1.3 were recorded twice. Each recording session consists of exactly 300 frames, resulting in a total of 7800 frames. Of these, every 10th frame was labelled for a total of 780 labelled frames, which were used for the preliminary model development.

When multiple persons are detected one person might be incorrectly detected in the background. While analysing the data the person that is not labelled as faulty is selected whenever possible. If a person is labelled as faulty, each joint is marked as in an unrealistic position.

An important factor in how well a model for HPE error detection can be trained on data is the balance of the dataset. In this case, the dataset is balanced by the error labels.

Full Body Error Distribution

In Figure 4-8 the error distribution of the Full body problem set can be seen. It can be observed that the dataset is not very balanced.

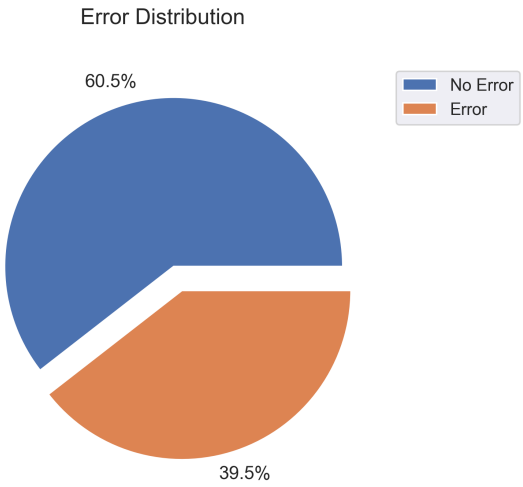


Figure 4-8: The distribution of Errors of the Full Body problem set. Of the 780 labelled frames, 308 are erroneous.

Half Body Error Distribution

Figure 4-9 shows a discrepancy between the error distribution of the lower body (65.4% Errors) and the upper body (32.4% Errors). The upper body is generally more stable and less error-prone than the lower body.

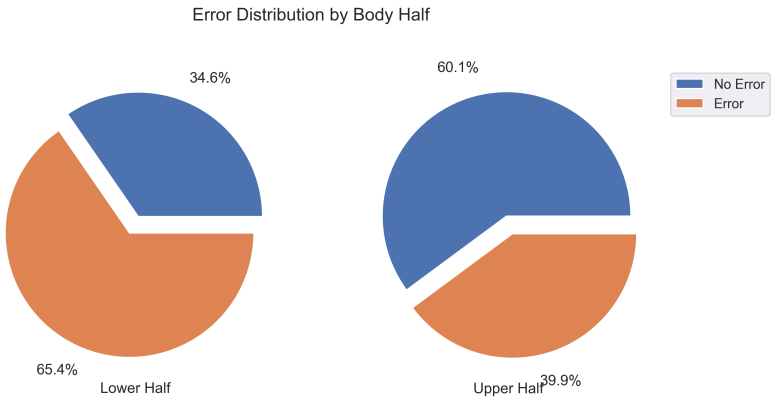


Figure 4-9: The distribution of Errors of the Half Body problem set. Of the 780 labelled frames, 511 and 311 are erroneous for the lower and upper body respectively.

Body part Error Distribution

The error distribution of each body part is shown in Figure 4-10. It can be observed that the errors of the body parts individually are very unbalanced. The left arm is the most error-prone body part with 24.1% of the joints being faulty. The torso is the least error-prone body part with 10.3% of the joints being faulty.

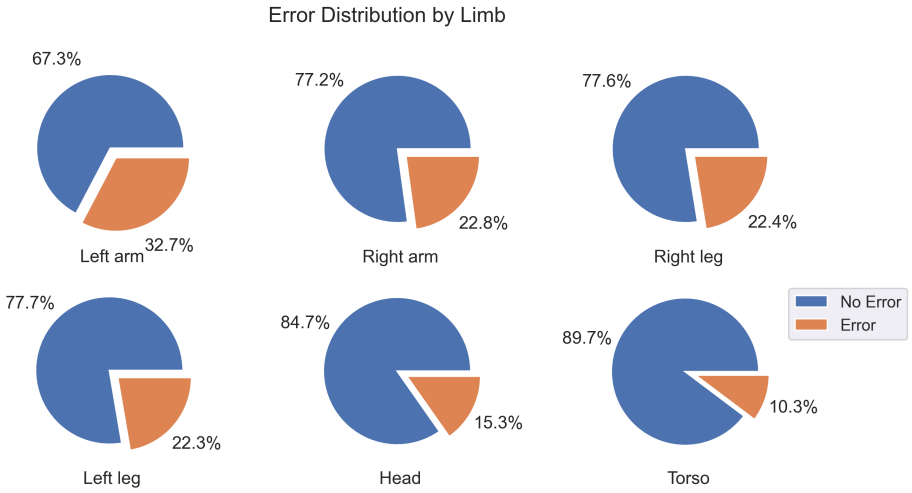


Figure 4-10: The distribution of errors of the Body Part problem set. The Left Arm is the most erroneous body part (255 errors), followed by the right arm (178 errors), the right leg (176 errors), the left leg (174 errors), and the head (119 errors). The least errors occur in the torso (80 errors in 780 frames).

Joint Error Distribution

Figure 4-11 shows that the major part of the errors that occur are errors with Label 2, i.e. the joint is detected at the wrong place. The second most common error is Label 1, i.e. the joint is not detected at all. The least common error is Label 3, i.e. the joint is detected in the approximate position of where another joint should be.

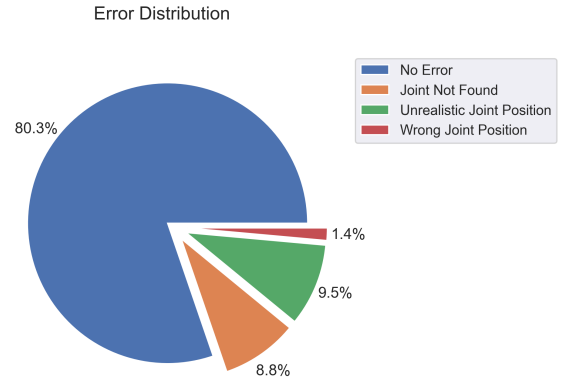


Figure 4-11: The distribution of each error class. The Right Ankle is the most erroneous joint (542 errors), followed by the left ankle (520 errors), the left hand (249 errors), the left wrist (183 errors), the right hand (169 errors), the right wrist (146 errors), the left hip (136 errors), the right hip (122 errors), the right knee (147 errors), the waist (117 errors), the left knee (127 errors), the right elbow (101 errors), the left elbow (77 errors), the head (75 errors), the neck (73 errors), the right shoulder (65 errors), the right collar (63 errors), the torso (63 errors), and the left collar (63 errors). The least errors occur in the left shoulder (38 errors in 780 frames).

Chapter 5

FESDModel - Preliminary Experiments for Model development

While there could be multiple approaches to fault estimation, a deep-learning approach using deep convolutional neural networks(CNN) has been chosen. The reason for this is that CNNs have shown to be very successful in many different fields, especially in computer vision tasks, such as image classification, object detection, and image segmentation.

We speculate that other possible solutions could be to use rule-based systems, which use inverse kinematics, or use frame-to-frame joint comparison to detect discrepancies, however, we expect that these are quite limited and might result in either too many false positives or false negatives, but these are speculations and need to be further investigated to form a clear understanding of alternative approaches for error detection. As no research has been found to actual error detection this is left as future work as such an investigation is a task in its own right.

The closest approach to error detection was proposed by Joao Carreira et. al for their Iterative Error Detection algorithm. However, in their work, the authors do not detect errors but rather determine the best change that can be applied to an estimated pose to achieve a better pose. Therefore, this approach cannot efficiently be used to determine an error in a scene, as a correction does not necessarily mean that there is an error.

Here two different models are proposed as they could in general be designed for this kind of task on a dataset consisting of three different modalities, RGB, Depth, and Pose Data.

5.1 Model architecture

For solving the problem of error estimation, two different model architectures are proposed. The first model architecture, FESDModelv1, is a deep convolutional neural network that uses the RGB, Depth, and Pose data as separate inputs. This model can be seen in Figure 5-1.

The second model architecture, FESDModelv2, utilises transfer learning to extract the features of the input data using a pre-trained model. The architecture of FESDModelv2 can be seen in Figure 5-2. Both models are trained to predict the error labels for each joint. The error labels are the same as the error labels used in the data labelling explained in

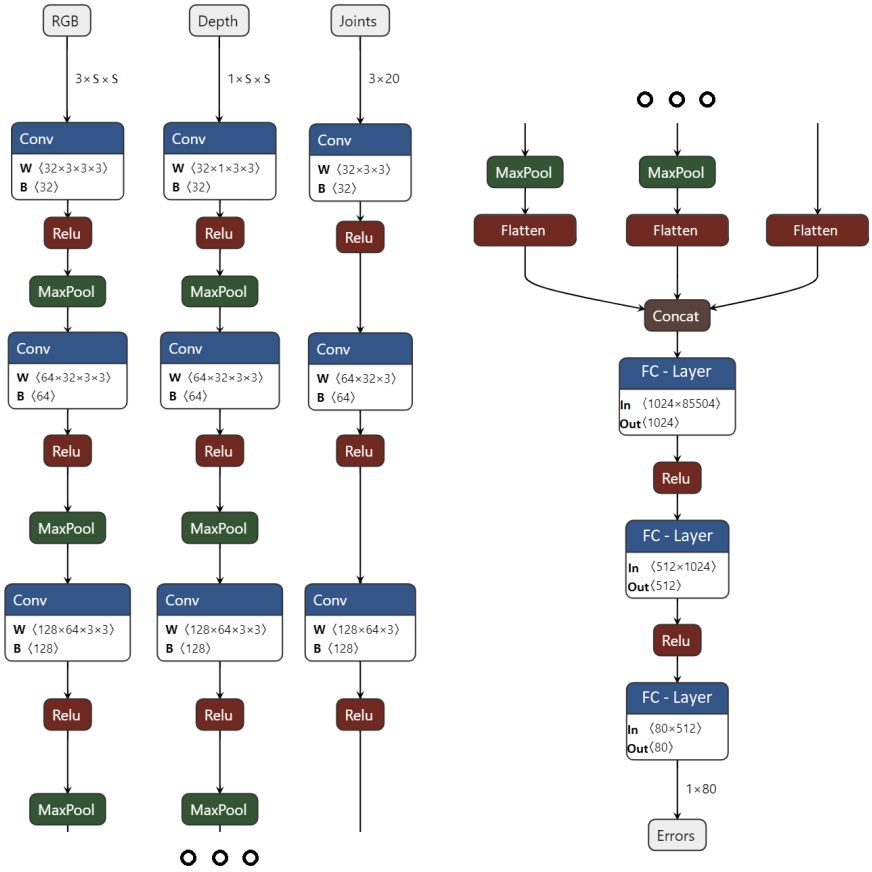


Figure 5-1: FESDModelv1 architecture with three different inputs; RGB, Depth and Joint data. With 'S' as the image size. After three convolutions the three streams are concatenated to be passed into three fully connected layers with ReLU activation functions. In this example network, the model calculates the Joint problem set, therefore the output is a 1D 40 tensor consisting of pairs of boolean values. Each of the 20 pairs corresponds to an individual joint.

Section 4.4. The fully connected layers of both networks use intermittent rectified linear unit (ReLU) activation functions to combat the vanishing gradient problem by passing only the values which are greater than zero into the next layer.

While FESDModelv1 uses the data as it is stored in the dataset, FESDModelv2 merges the data into a single RGB image. This is done using the feature extractor, which is originally trained on RGB images. The data is merged by assigning each modality to a seperate channel of an RGB image. The RGB image is transformed into greyscale and assigned to the red channel of the RGB input image, the depth image is scaled to a value between 0 and 255 and assigned to the green channel, and the joint coordinates are depicted as white squares on an image, aligned to both the RGB and Depth image, assigned to the blue channel.

In total eight models were developed and trained. Four models were trained using FESDModelv1 and four models were trained using FESDModelv2. Each of the models corresponds to the problem sets A-D, as introduced in Section 4.5. And the model for each respective problem set is trained using both FESDModelv1 and one model for each problem set is trained using FESDModelv2.

Consequently, the output of the models varies depending on the problem set. Based on the problem sets, their respective problem areas and the error labels as discussed in Section 4.4, the Full Body, Half Body, Body Part, and Joint problem sets have an output vector of size 2, 4, 12, and 40 respectively. While more detailed information exists about the error for the Joint problem set, it was decided that the simplified "Error"/"No Error" label is being predicted.

FESDModelv2 uses a neural network which was pretrained on ImageNet as a feature extractor. Multiple candidate networks have been compared, which can be seen in Figure 5-3. One of the target applications of the model is to be used in a real-time application so that error handling can be conducted. Consequently, a lightweight model which does not impact the performance much is preferred. Therefore, the models are compared by the number of floating-point operations (FLOPS) to their Accuracy on ImageNet-1K. Table 5.1 shows the top 5 models according to their accuracy and performance.

Table 5.1: The top 5 models according to their accuracy and performance. The models are sorted by their GFLOPS and their Top-5 Accuracy(Source: https://pytorch.org/vision/main/models.html on 08/05/2023). The models are EfficientNet V2 S, ConvNeXt Base, EfficientNet B6, Swin V2 B, and EfficientNet V2 M.

Weight	Acc@1	Acc@5	Params	GFLOPs
EfficientNet V2 S	84.228	96.878	2.15×10^7	8.370
ConvNeXt Base	84.062	96.870	8.86×10^7	15.360
EfficientNet B6	84.008	96.916	4.30×10^7	19.070
Swin V2 B	84.112	96.864	8.79×10^7	20.320
EfficientNet V2 M	85.112	97.156	5.41×10^7	24.580

EfficientNet v2 was chosen since it proved to be the most performant while being the most accurate of the networks that were analysed. In particular, the small variant with 2.15×10^7 Parameters and a Top-1 Accuracy of 84.228%[27]. While EfficientNet V2 M outperforms EfficientNet V2 S in terms of Top-1 Accuracy, the number of additional parameters needed to achieve a better accuracy outway the performance bonus that EfficientNetv2 S

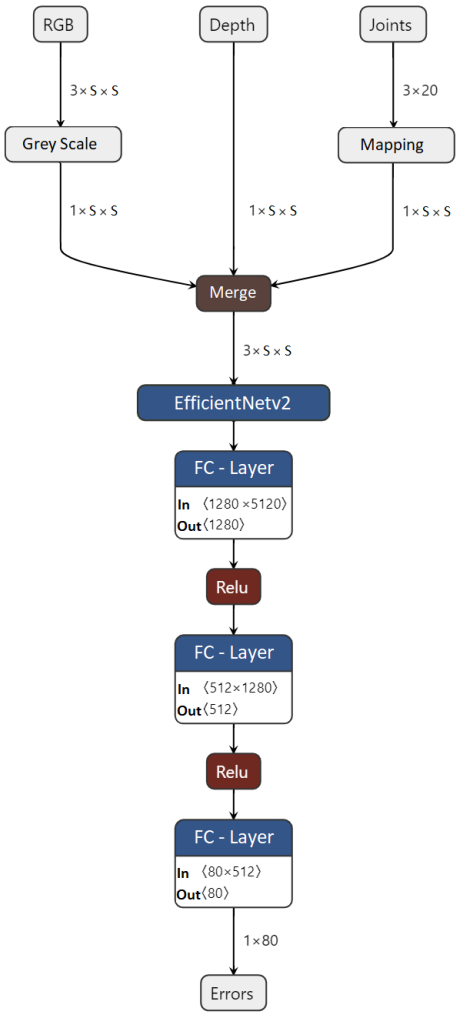


Figure 5-2: FESDModelv2 architecture with transfer learning. The input is merged into a single RGB image and passed into a feature extractor. With 'S' as the image size. The feature extractor is a pre-trained EfficientNet v2 S. The output of the feature extractor is passed into two fully connected layers with ReLU activation functions. In this example network, the model calculates the Joint problem set, therefore the output is a 1D 40 tensor consisting of pairs of boolean values. Each of the 20 pairs corresponds to an individual joint.

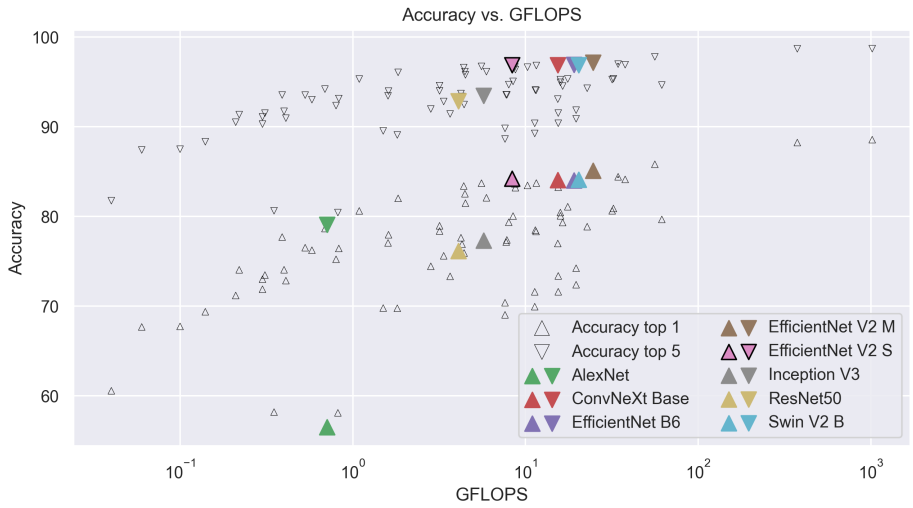


Figure 5-3: The comparison of different networks by their GFLOPS and their Top-5 Accuracy. The models are sorted by their GFLOPS and their Top-5 Accuracy(Source: https://pytorch.org/vision/main/models.html on 08/05/2023). The models are EfficientNet V2 S, ConvNeXt Base, EfficientNet B6, Swin V2 B, and EfficientNet V2 M. Additionally, AdamNet, ResNet-50 and Inception-v3 are added as a reference.

brings with it. EfficientNetv2 is a convolutional neural network, which optimises training speed and parameter efficiency and improves upon EfficientNet[28]. The main focus of EfficientNet is the scaling of the model in width, depth and resolution of the input image.

5.2 Data preparation

To successfully train FESDModel three steps are taken before training can begin, data augmentation, data merging, and data balancing. The data augmentation is done to ensure that the model is robust to different variations in the data. The data merging is done to combine the different modalities into a single tensor. Since the dataset is unbalanced, the data balancing is done to ensure that the model is not biased toward any particular error label.

The finished data preparation pipeline for FESDModelv1 and FESDModelv2 can be seen in Figure 5-4.

The joints are stored within a JSON file containing the coordinates of each joint in 2D and 3D. To process the 2D joint data is drawn on an image that has the same dimensions as the RGB and Depth image for FESDModelv2. For FESDModelv1 the position of each joint relative to the waist joint is passed into the network.

5.2.1 Data augmentation

Four different augmentations are applied to the data to generalise the data. The first augmentation is flipping the data. The RGB image, the depth image, and the joint image are flipped horizontally. Furthermore, the ground truth data is flipped, as labels refer to the left or right side of the body, which would no longer coincide with the data that is passed into the network.

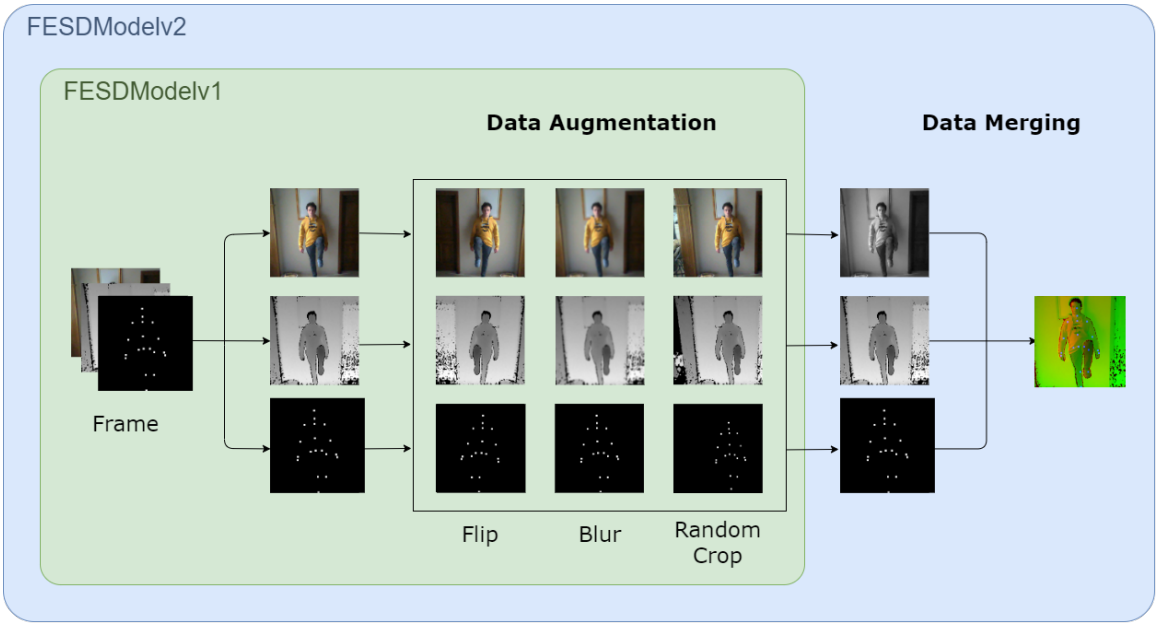


Figure 5-4: The three modalities are seperately randomly flipped, Blurred, and Cropped, i.e all three modalities are flipped, blurred, or blurred together in the same way. For FESDModelv1 the separate modalities are passed into the network. For FESDModelv2, the RGB image is transformed into a greyscale image and all three modalities are merged into a single RGB image.

Additionally, the images are cropped at random while keeping the positions of the joints and a margin around the joints visible. This ensures that the model is robust to different positions of the user in the image.

Finally, Gaussian noise is applied to the RGB image and the depth image. This further improves the robustness of irregular data.

The augmentations can be seen in Figure 5-4 where they are applied to a sample frame from the dataset. The augmentation methods were chosen based on their suitability for HPE applications.

5.2.2 Data balancing

In the ordinary case, HPE is not meant to produce faulty results. The exercises are selected for their challenges and are expected to produce faulty results. However, this does not automatically produce a balanced dataset. In Section 4.7, the statistics of the dataset are shown. Most notably for the problem set *Half* and *Full*, in figures 4-9 and 4-8, where the error label *No Error* is overrepresented it can be seen that the dataset is imbalanced.

To balance the dataset, frames are sampled using a Weighted Random Sampler for each batch of the training. The weights for the samples are calculated based on the occurrence of the error labels in the dataset. While only considering the whole body as a single object, the calculation of the weights is simple. For each frame, the error label is counted and the inverse of the count is used as the weight for the frame. This ensures that the model is not biased toward any particular error label by oversampling the frames which contain an error.

However, for the other problem sets the calculation of the weights is more complex. In the other problem sets each frame contains an error for each area, e.g. when considering the

Half-Body problem, the upper and lower body 2 errors. To successfully balance the dataset for each area four weights would need to be created and balanced, i.e. the upper and lower body have an error the upper body is faulty and the lower body is not, etc. This would oversample some frames while undersampling others. In the other problem sets this is far more visible. Therefore, it was decided to consider the sum of erroneous joints per frame as a balancing factor. This means that frames that have the same number of erroneous areas are weighed the same. For example, resulting in 20 different weights for the Joint problem set, i.e. none of the joints are faulty, one joint is faulty, etc. If each joint combination was weighted individually this would result in over 1e6 different weights, i.e. no joint is faulty, the left foot is faulty, the left and right foot are faulty, etc.

5.3 Experimental Setup

To train the models the data has to be passed into the network so that it can predict a value. Based on that value a loss is calculated which is used to adapt the weights of the networks. In the case of both FESDModelv1 and FESDModelv2 *cross entropy loss* is used. In cases where the problem set contains more than one problem area, i.e. all problem sets except the Full Body problem set, the cross-entropy loss is calculated for each object or area separately and an average cross-entropy loss is calculated.

Initially, low-resolution images of 64x64 pixels were used for training to minimise the prediction time and the training time. However, this proved to negatively impact the performance of FESDModelv2. Therefore, more iterations of the training were run where the images are rescaled to a resolution of 200x200 pixels. This resolution was chosen to limit training time and optimise performance while keeping a high resolution to improve the accuracy of the model. No extended investigation into the resolution of the input image has been conducted and remains subject to future research.

The models that were trained on low-resolution images were trained for 50 epochs. Due to limited time, the models were only trained for 20 epochs on the images with a higher resolution.

Both networks are trained using the Adam optimiser, as described by Diederik P. Kingma and Jimmy Ba[29], with an initial learning rate of 0.00005 with learning rate decay.

To improve the performance of the training process Cuda is used.

Chapter 6

Results

In this chapter, the results of the experiments are presented. The results for each different problem sets are presented separately. Note that the HPE error detection task can be seen as a classification task, where something classified as an *error*, e.g. a joint classified as an error, is considered a positive classification and *no error* is considered a negative classification. Therefore, a result is considered as a "True Positive", if the ground truth label of a joint is equal to "Error" and the predicted value of that joint is also equal to "Error", etc. Models which suffered from neural collapse are omitted to reduce confusion.

6.1 FESDModelv1 Results

Initially, FESDModelv1 was trained on low-resolution 64x64 pixels images, to improve training and predicting speeds. Further experiments were conducted on higher resolution images. In the next subsections, we present the results of both training variations. Note that the models trained on the low-resolution images were trained for 50 epochs, while when training on images with a higher resolution, the models were trained for 20 epochs due to time limitations.

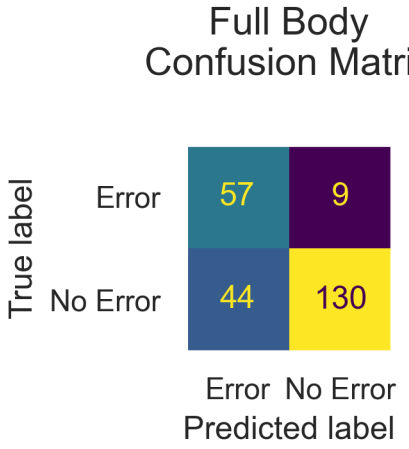
6.1.1 Results for low-resolution images

After training the FESDModelv1 on all four problem sets respectively, we obtained the results, as listed in Table 6.1. These are the performance results on the test data, after 50 epochs of training on low-resolution images.

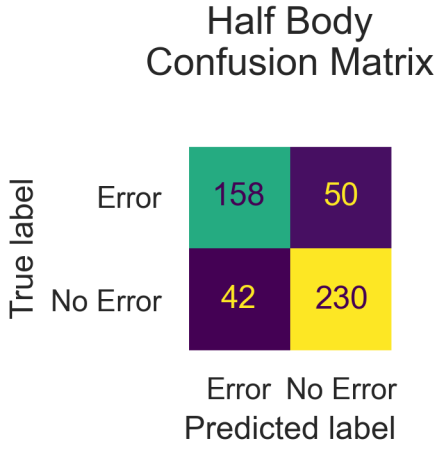
Note that the results of the Joint problem set have been omitted since it suffered from neural collapse. Here, the low resolution of the input images prohibited obtaining a reasonably performing FESDModelv1 model.

Table 6.1: The test results of FESDModelv1 after 50 epochs of training on the Full Body, Half Body and Body Parts dataset. Showing the Percentage of Positive Guesses (PPG), the Accuracy (Acc), the F1 Score, and the Cohen's Kappa Coefficient (κ).

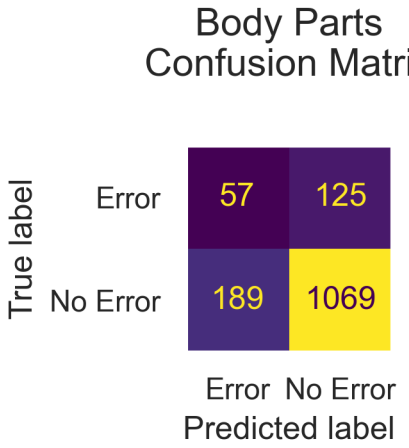
Problem Set	PPG	Acc	F1	κ
Full Body	0.42	0.78	0.68	0.52
Half Body	0.42	0.81	0.77	0.61
Body Parts	0.17	0.78	0.27	0.14



(a) Full Body Problem Set



(b) Half Body Problem Set



(c) Body Parts Problem Set

Figure 6-1: The confusion Matrices of FESDModelv1 for the (a) Full Body, (b) Half Body, and (c) Body Parts problem sets trained on input images of 64x64 pixels resolution.

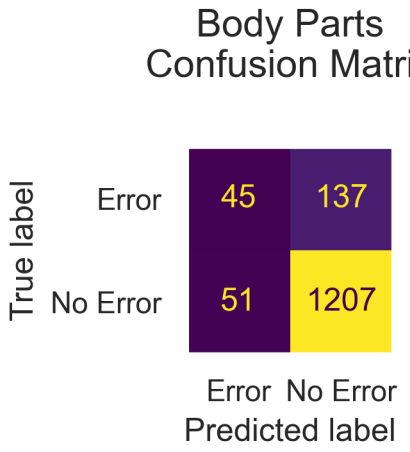
6.1.2 Results for higher-resolution images

Table 6.1 shows the results of the testing after 20 epochs of training the models on the Body Parts and Joints problem set on higher-resolution images. The models of each of the two problem sets perform slightly better than the low-resolution images. However, a direct comparison is difficult to make, since the problem sets that are predicted by FESDModelv1 on low resolution images suffer from neural collapse on the higher resolution images and vice versa. When comparing the Body Parts problem set FESDModelv1 performs better, when trained on higher resolution images.

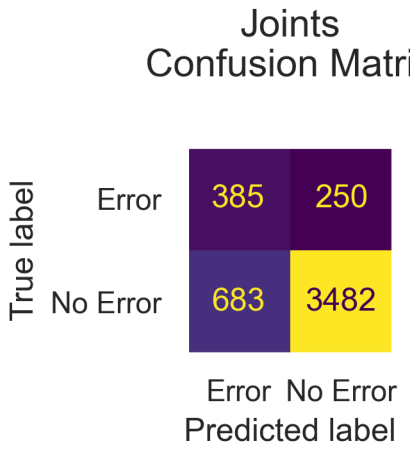
Table 6.2: The test results of FESDModelv1 after 22 epochs of training on the Body Parts and Joints dataset. Showing the Percentage of Positive Guesses (PPG), the Accuracy (Acc), the F1 Score, and the Cohen's Kappa Coefficient (κ).

Problem Set	PPG	Acc	F1	κ
Body Parts	0.07	0.87	0.32	0.26
Joints	0.22	0.81	0.45	0.34

FESDModelv1 could be trained on the Joint problem set when it is trained with higher resolution input images, resulting in a model that obtains an F1-Score of 0.45. The model might perform better when trained for longer, however, due to time limitations this was not possible and is left for future work.



(a) Body Parts Problem Set



(b) Joints Problem Set

Figure 6-2: The confusion matrices of FESDModelv1 for the Body Parts, and Joints problem sets.

6.2 FESDModelv2 Results

Similar to FESDModelv1, FESDModelv2 was first trained on 64x64 pixels input images and subsequently on 200x200 pixels images.

6.2.1 Results for low-resolution images

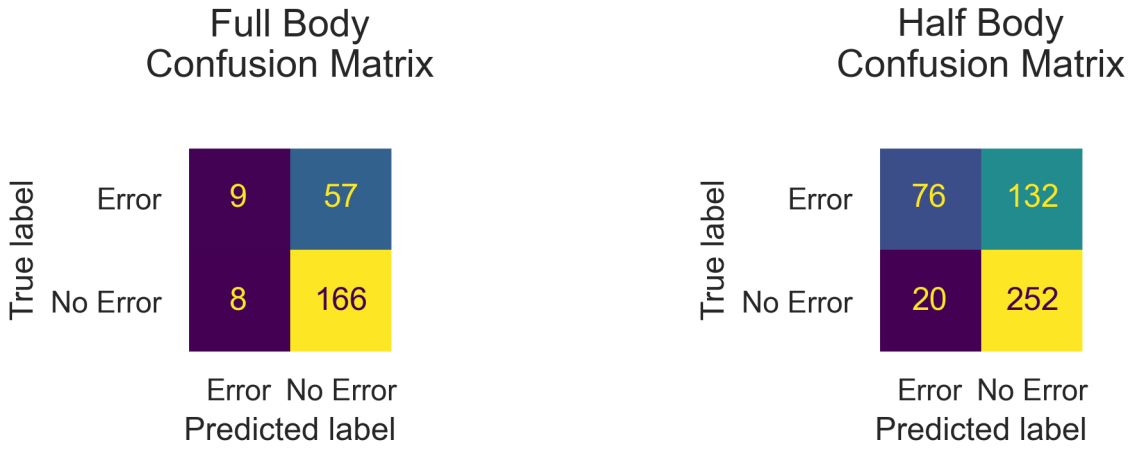
The results of the testing after 50 epochs of training for FESDModelv2 on low-resolution images as listed in Table 6.3.

The results of the Body Part and Joint problem set have been omitted since the low resolution of the input images prohibited reasonable training for the datasets.

Table 6.3: The test results of FESDModelv2 after 50 epochs of training. Showing the Percentage of Positive Guesses (PPG), the Accuracy (Acc), the F-β score calculated with $\beta = [1, 0.5, 2]$ (F1, F0.5, F2 respectively) and the Cohen's Kappa Coefficient (κ).

Problem Set	PPG	Acc	F1	κ
Full Body	0.07	0.73	0.22	0.12
Half Body	0.20	0.68	0.50	0.31

Note that, FESDModelv2, when trained on low resolution images obtains F1 Scores of 0.22 and 0.50, respectively, which are not good results as a trivial model that always predicts positive values would achieve already better F1 Scores of 0.43 and 0.60, respectively.



(a) Full Body Problem Set

(b) Half Body Problem Set

Figure 6-3: The confusion Matrices of FESDModelv2 for the Full Body and Half Body problem sets trained on input images of 64x64 pixels resolution.

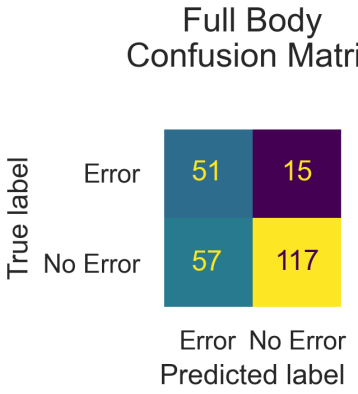
6.2.2 Results for higher resolution images

The results of the testing after 20 epochs of training for FESDModelv2 on 200x200 pixels resolution images of the Full Body, Half Body, Body Parts and Joints problem sets, respectively, as listed in Table 6.4.

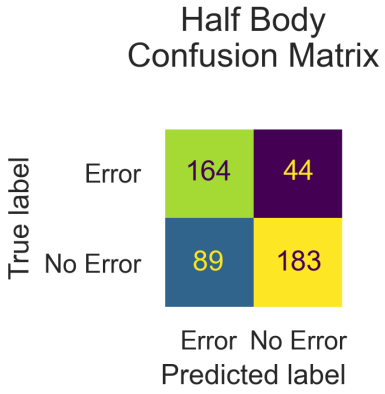
Table 6.4: The test results of FESDModelv2 after 22 epochs of training on all datasets. Showing the Percentage of Positive Guesses (PPG), the Accuracy (Acc), the F1 Score, and the Cohen's Kappa Coefficient (κ).

Problem Set	PPG	Acc	F1	κ
Full Body	0.45	0.70	0.59	0.37
Half Body	0.53	0.72	0.71	0.45
Body Parts	0.06	0.92	0.09	0.06
Joints	0.11	0.90	0.57	0.51

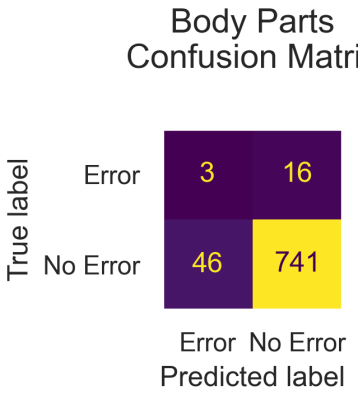
FESDModelv2 achieves better F1-Score FESDModelv1 for the Joint problem set of all models when trained on images with a higher resolution. Furthermore, very promising results are achieved on the Half Body problem set with an F1-Score of 0.71. Finally, for the Full and Half Body problem sets higher F1 scores are obtained when using high resolution images.



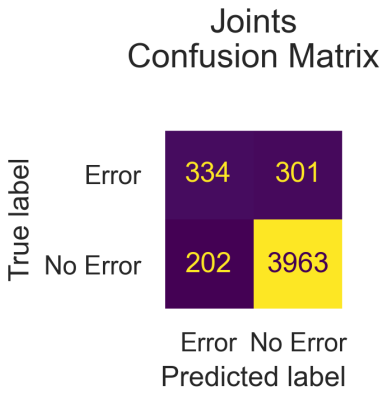
(a) Full Body Problem Set



(b) Half Body Problem Set



(c) Body Part Problem Set



(d) Joint Problem Set

Figure 6-4: The confusion Matrices of FESDModelv2 for the different problem sets trained on input images of 200x200 pixels resolution.

Chapter 7

Conclusion

In this thesis, a benchmark dataset for the evaluation of HPE error detectors has been constructed. HPE error detection is important for the improvement of the user experience in several applications such as serious gaming as used by SilverFit.

The following important research questions were addressed; (A) What errors occur during HPE and how can these errors be reproduced in a controlled manner? (B) How can a dataset of multiple modalities be captured, to analyse if the previous observations about problems during pose estimation are correct? (C) How can the previously captured and labelled dataset be used to train a model using multi-modal data to determine if a joint is faulty or not?

A Common problems of HPE were analysed and exercises were designed that cause challenges for HPE in a controlled manner. It was found that lighting has a high impact on the performance pose estimation, as well as the posture of the user, and the visibility of the head.

B Using these exercises a dataset, FESDDataset, was captured and labelled using a custom tool for multi-modal stream capture and labelling for HPE error detection. Four different problem sets were captured and labelled that encompass different levels of challenges for HPE.

C Finally, two different models, FESDModelv1, and FESDModelv2 were proposed with which preliminary HPE error detection experiments were conducted on the four datasets. As input the DNN models used RGB, Depth and pose Data. FESDModelv1 extracts the features of each modality individually, whereas FESDModelv2, combines the RGB, depth, and pose data into a single RGB image, which is passed into a pre-trained model for feature extraction.

Preliminary experiments show that both models are able to attain positive results on several of the proposed problem sets. In our experiments FESDModelv2 did not perform better than FESDModelv1 on comparable problem sets, except for the Joint problem set with high resolution images, however, further experiments are necessary in order to be able to draw any final conclusions.

In conclusion, the results of the preliminary experiments are promising and show that the development of a model for error detection using RGB, Depth and Pose Data is a viable possibility.

7.1 Contributions

In this thesis, Exercises were proposed, which challenge HPE at varying difficulty levels. Furthermore, a HPE dataset construction tool, FESDData was designed and implemented, which allows the capture of RGB, Depth and Pose data simultaneously and allows for the labelling of errors. Using that tool FESDDataset was captured and constructed which contains 300 frames for 13 Exercises which were recorded twice, resulting in a total of 7800 frames of multimodal HPE and RGB-D data. Finally, preliminary experiments were conducted to show the viability of model development for HPE error detection using the novel datasets.

The code of this thesis is available on GitHub (https://github.com/LeonardoPohl/FESD). The repository is divided into two major parts, FESDData, which contains the C++ implementation of FESDData, FESDModel, which contains the implementation of the model, FESDModelv1 and FESDModelv2, as well as the Jupyter notebooks that were used to evaluate the dataset, to train and evaluate the model. FESDDataset, as well as the trained models, are available on request.

7.2 Future work

To further improve the dataset and subsequently, the model, FESDData and FESDDataset could be expanded to include the accurate position of the joints in the image. This would allow for the use of the dataset for error correction.

To improve the quality of the FESDModel, more data needs to be collected and labelled in different settings. The current dataset is limited to a single room with a single camera. To improve the model, the dataset needs to be expanded to include different scenes, different pieces of clothing and different camera angles. These additions might prevent the model from overfitting. Additionally, the model could be improved by using a different backbone, which is not as performant but more accurate, such as EfficientNet v2 M.

Different architectures for a model could also be imagined. For example, the development of multiple smaller models, which are designed with a specific problem area in mind might offer an improvement over the existing models.

7.2.1 Possible applications

FESD might find several different areas of application in the future. Firstly, the trained model can be used to assist in developing games and other applications that utilise HPE. In its simplest application, it may be used to warn users of possible errors when an error is detected by FESDModel. In more advanced cases the information provided by the model could be used to attempt to fix joints using for example joint position interpolation or prediction rather than using the faulty joint. Moreover, multiple human pose estimators could be considered resulting in an overall more robust HPE.

Furthermore, if the model proves to have high accuracy for a specific use case, it could be used to train a better pose detector in the same way as it is proposed by João Carreira et al.[10].

The dataset and the dataset recorder may also be used to further the development of FESDModel and it can also find application in other areas such as recording datasets for action recognition. The dataset in and of itself can be used for action detection. The

exercises are predefined and can be recorded and automatically labelled by FESDData, thereby making it easy to record large amounts of data without requiring manual labelling.

References

[1] L. Kumarapu and P. Mukherjee, "AnimePose: Multi-person 3D pose estimation and animation", *Pattern Recognition Letters*, 2020.

[2] K. Chen, P. Gabriel, A. Alasfour, C. Gong, W. K. Doyle, O. Devinsky, D. Friedman, P. Dugan, L. Melloni, T. Thesen, D. Gonda, S. Sattar, S. Wang, and V. Gilja, "Patient-Specific Pose Estimation in Clinical Environments", *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 6, pp. 1–11, 2018.

[3] B. Preim and M. Meuschke, "A survey of medical animations", *Computers and Graphics*, vol. 107, Sept. 2022.

[4] F. Kröger, *Automated Driving in Its Social, Historical and Cultural Contexts*, pp. 41–68. Springer, May 2016.

[5] J. Stenum, K. M. Cherry-Allen, C. O. Pyles, R. Reetzke, M. F. Vignos, and R. T. Roemmich, "Applications of Pose Estimation in Human Health and Performance across the Lifespan", *Sensors (Basel, Switzerland)*, vol. 21, 2021.

[6] M. Tölgyessy, M. Dekan, and L. Chovanec, "Skeleton Tracking Accuracy and Precision Evaluation of Kinect V1, Kinect V2, and the Azure Kinect", *Applied Sciences*, vol. 11, p. 5756, June 2021.

[7] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[8] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand Keypoint Detection in Single Images using Multiview Bootstrapping", in *Computer Vision and Pattern Recognition*, 2017.

[9] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", in *Computer Vision and Pattern Recognition*, 2017.

[10] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik, "Human Pose Estimation with Iterative Error Feedback", *Computer Vision and Pattern Recognition*, 2015.

[11] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on $\mathcal{X}$-transformed points", *Computer Vision and Pattern Recognition*, 2018.

[12] B. Seddik, S. Gazzah, and N. Essoukri Ben Amara, "Human-action recognition using a multi-layered fusion scheme of Kinect modalities", *IET Computer Vision*, vol. 11, no. 7, pp. 530–540, 2017.

[13] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite", in *Conference on Computer Vision and Pattern Recognition (Computer Vision and Pattern Recognition)*, 2012.

[14] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 1325–1339, July 2014.

[15] M. Fisch and R. Clark, "Orientation Keypoints for 6D Human Pose Estimation", *Computer Vision and Pattern Recognition*, 2020.

[16] Y. Liu, "Contour Model and Robust Segmentation based Human Pose Estimation in Images and Videos", *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 8, pp. 1–10, Mar. 2015.

[17] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods", *Computer Vision and Image Understanding*, vol. 192, p. 102897, 2020.

[18] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D Human Pose Estimation: New Benchmark and State of the Art Analysis", in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2014.

[19] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context", 2014.

[20] L. Sigal, A. Balan, and M. Black, "HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion", *International Journal of Computer Vision*, vol. 87, pp. 4–27, Mar. 2010.

[21] S. An, Y. Li, and U. Ogras, "mRI: Multi-modal 3D Human Pose Estimation Dataset using mmWave, RGB-D, and Inertial Sensors", *Computer Vision and Pattern Recognition*, 2022.

[22] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera", in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 755–762, 2010.

[23] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real-Time Human Pose Tracking from Range Data", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database", in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[25] C. Lee, "The roc space for a "better" and "worse" classifier.". Wikipedia, June 2018.

[26] M. McHugh, "Interrater reliability: The kappa statistic", *Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB*, vol. 22, pp. 276–82, Oct. 2012.

[27] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training", *Computer Vision and Pattern Recognition*, 2021.

[28] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks", *Computer Vision and Pattern Recognition*, 2020.

[29] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2017.