Abstract

From early punch cards to the more recent voice activation, with each new technology, the interaction between humans and computers has become more natural and unobtrusive. One of these newer advances is the interaction with computers based on visual input. Thanks to faster and more available hardware, we can analyse video streams in real time and use the information to enable the interaction between humans and computers. However, this interaction is not always as accurate as desired.

In the scope of this thesis, a method is developed that is capable of capturing and labelling multi-modal data, **F**ault **E**stimator for **S**keleton **D**etection **Data** processor (FESDData). Using the dataset that is recorded and labelled using FESDData, I developed a model which aims to detect if an error occurs, **F**ault **E**stimator for **S**keleton **D**etection **Model** (FESDModelv2). FESDModelv2 is an improvement over an older model that was also developed in the scope of this thesis. The new version uses transfer learning to extract the features from the data. This thesis is written in collaboration with SilverFit, a serious gaming company that develops video games for rehabilitation purposes. SilverFit deals with unnatural poses due to injury or old age in a lot of cases.

The result of my research shows that the collected and labelled data is not enough to create a generalised model for error detection. There are too many parameters However, the developed method for data collection might still prove useful in future applications. This opens ways to improve the quality of the interaction between humans and computers by improving the robustness of human pose estimation.

Contents

List of Figures

Chapter 1

Introduction

Human pose estimation aims at detecting the pose or skeleton of a person based on visual information only. It finds many applications, from games to medical applications[20, 7, 24]. However, human pose estimation is a difficult task that is prone to errors[32]. These errors can be caused by different factors, such as the environment, the camera, and the person. These errors can cause the joints of the pose to be in the incorrect position or missing. This can cause the pose estimation to be incorrect, and therefore, the human-computer interaction will be hampered. The goal of this thesis is to develop a method that allows for the detection of these errors such that the pose information can either be improve on them or that or be handled accordingly.

One of the limiting factors for human pose estimation is the user itself. If the posture is not as the 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, as is 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 human pose estimation to detect the pose of the player and use it to control the game to make exercise more enjoyable while promoting activity. This thesis is written in colaboration with SilverFit and aims at improving the human pose estimation by giving featback on the errors in their games.

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

In this chapter, the research question that is answered in this thesis is discussed. Furthermore, the procedure with which the research question is answered is laid out. Finally, the fundamentals of human pose estimation and the metrics that are used to analyse the results are discussed, and related work is presented.

1.1 Research question

A major problem with human pose estimation is that it is not possible to tell if the joints of the pose are faulty or not. Using faulty joints can decrease the efficacy of the training effect of the developed games and can make them very frustrating to use and develop. A

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

joint is considered faulty if it is not in the incorrect position, i.e. the distance from the actual position is greater than a chosen threshold, or if it missing from the skeleton.

In this thesis, first, the problems that occur during human pose estimation are analysed. The aim is to find which problems are the most common and which joints are most affected by the errors. Additionally, exercises are designed to emulate different scenarios with ever-increasing difficulty. This will help give an overview of the issues related to human pose estimation and help develop ways to detect these issues.

Once the issues and error sources that occur during human pose estimation are analysed, a method is developed to capture the camera stream in a way that allows the labelling of the data according to the exercise and environment it was captured in. This allows for the creation of a dataset that can be used to train a model to detect faulty joints.

Using the dataset that is developed we train a model. Using the model we answer the research question posed; "Is it possible to train a model using multi-modal data to determine if a joint is faulty."

1.2 Process Pipeline

In the scope of the thesis, three steps were defined that are necessary to achieve the goal of the thesis. These steps are the analysis of the problem, the data collection and processing, and the model development.

During the analysis of the problem, different factors are discussed which might influence the human pose estimation. Based on these factors exercises are designed to emulate different scenarios with different difficulties. This is discussed in Section 2.

The data collection and processing are discussed in Section 3.1. The data is captured using a custom tool, FESDData, that was developed for this thesis. The tool allows capturing predefined exercises and labelling them with error labels. The data is then processed and stored in a custom format.

Using the collected dataset a data loader and model are derived. The data loader performs the necessary preprocessing steps and the model is trained to detect faulty joints. This is discussed in Section 4.

Finally, the model is evaluated on a test set and the results are discussed in Section 5.

1.3 Fundamentals

In this section, the fundamentals that are necessary to understand the thesis are discussed. These fundamentals are human pose estimation, the fundamentals of machine learning, and the evaluation metrics that are used to evaluate the results.

1.3.1 Human pose estimation

There is wide variaty 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.

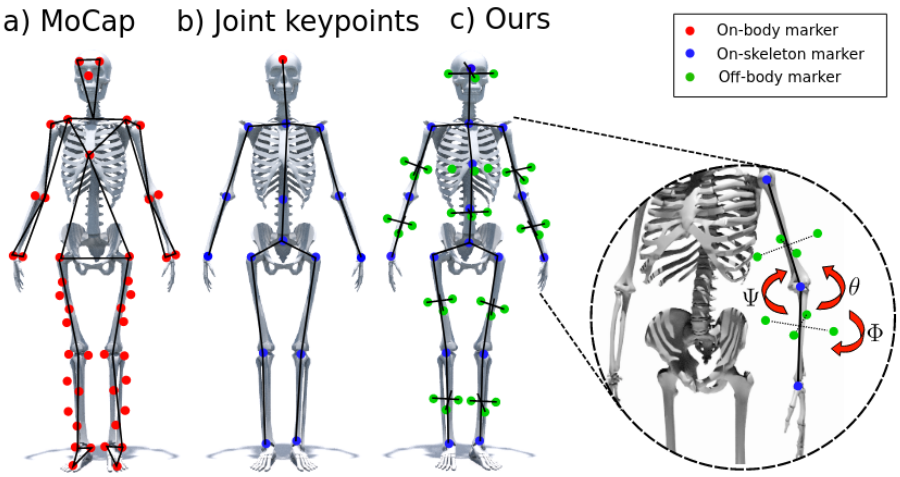


Figure 1-1: Example of a human pose captured with different methods. (a) A captured skeleton using MoCap which $REPLACE_WE$ do not focus on in this report. (b) A traditional human skeleton representation. (c) A pose representation that includes the orientation of the joints as well as the bones as presented by Martin Fish and Ronald Clark[12]

Pose visualisation

Different methods to capture the pose of a human have been developed. 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, but the most common skeleton is made up of 17 joints and 16 bones, as can be seen in Figure 1-1. The joints are usually labeled with a number, which is used to identify the joint in the output of the pose estimation. The representation of a joint in the data varies, but it is usually a 2D or 3D point in space. In some cases, an additional joint representation is provided with a keypoint orientation that enables the clear representation of all degrees of freedoms joints have[12]. 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.

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[22]. 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. This is useful when the skeleton is not able to determine the shape of the body, for example, when the person is wearing a coat or a jacket. This is also used for some games developed by SilverFit.

Finally, the third way to represent a human pose is with a three-dimensional volume. 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. The three different

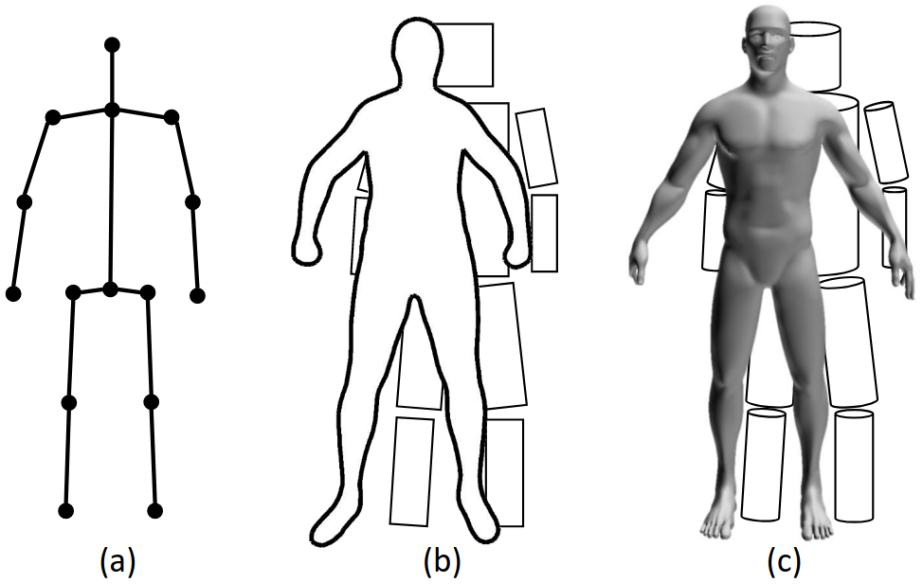


Figure 1-2: Different representations of the human pose. (a) Kinematic representation. (b) Contour representation. (c) 3D volume representation. [9]

representations of the human pose can be seen in Figure 1-2.

Data sources

The data that is acquired influences the way the way the human pose can be estimated. The most common data source is RGB images or videos. RGB videos are any videos that are captured with normal cameras that record the color 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 pose estimation algorithms from RGB data. Some of the most common datasets are the MPII Human Pose Dataset[2], the COCO dataset[21], and HumanEva-I dataset[28].

Additionally, some datasets are captured with depth cameras. Depth cameras are cameras that can record, additionally to the color channels, the depth of the scene. Different methods to acquire depth data have been developed. The most commonly used depth cameras emitt an infrared light pattern to measure 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[1], and the Human3.6M dataset[17].

Finally, there are also methods of human pose estimation 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[14], and the EVAL dataset[15]. However, any RGBD 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.

Applications

Human pose estimation finds application in many different fields. In this chapter some of the most common applications of human pose estimation are mentioned.

Gaming and entertainment One of the most common applications of human pose estimation. Games can use human pose estimation 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. The Kinekt used a depth camera to track the movement of the player and used this information to control the games. [**?**, Citation needed]

However, since human pose estimation requires space and a good camera, it is not as wide spread as conventional input peripherals.

Autonomous Driving Autonomous driving has been in development ever since humans replaced horses with cars[19]. However, the development of autonomous driving has been very slow. The main reason for this is that autonomous driving requires a lot of information about the environment. This information is usually provided by sensors that are installed in the car. However, sensors alone do not always suffice. 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.

Animation To exactly emulate human movements in animation, animators can either manually move the joints of a digital skeleton or they can use real human[2] actors to provide the movement for them. The manual creation of realistic movement is oftentimes very time-consuming and also error-prone. Therefore, animators often use real human actors to provide the movement for them. This provides animators with a skeleton and movement which is accurate and does not include human error. In large production studios, this is often done with motion capture or MoCap.

MoCap is a technique that uses cameras to capture the movement of a human actor. The cameras are placed around the actor and record the movement of the actor. The actor usually wears a suit that is covered with markers. These markers are used to determine the position of the actor. To reduce the amount of occlusion of the markers a large number of cameras are used. This allows the cameras to capture the movement of the actor from different angles. However, this also increases the price of development. In cases where MoCap is not a viable option, animators can use human pose estimation to estimate the pose of a human actor using cheaper RGB cameras or RGBD cameras.

Security Another application of human pose estimation is the detection of annomalous behaviour. The detection of such behaviour can be used to prevent security incidents in public spaces.

[2]Or animal

Healthcare To aid in the rehabilitation of patients as well as to improve the quality of life of elderly people, human pose estimation can be used to detect the pose of a person and provide feedback on the pose of the person during exercises.[7]

1.3.2 Machine Learning

Im not sure into how much detail I should go here.

1.3.3 Evaluation metrics

Precision, Recall, Accuracy and F1-Score

Cohens Kappa

Cross-Entropy Loss

Confusion Matrix

Percentage of positive Guesses

1.4 Related Work

1.4.1 Human Pose Estimation

RGB Pose Estimation

While OpenPose developed Hand Pose[29] and also Multi-Person Human Pose Estimation [4], $REPLACE_OUR$ main focus lies on their most recent pose estimator [3] and their CNN network [33]. Openpose uses affinity fields. The affinity fields are a set of 2D Gaussian distributions that are used to estimate the pose of a human. The affinity fields are used to estimate the pose of a human by estimating the probability of a joint being in a certain location. The probability of a joint being in a certain location is calculated by summing the probability of the joint being in that location for each of the Gaussian distributions.

RGBD Pose Estimation

1.4.2 Action Detection

1.4.3 Fault Detection

Human Pose Estimation using Iterative Error feedback. [5]

Chapter 2

Challenges in Human Pose Estimation

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

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

2.1 Environment

The environment in which the data is recocorded can be an issue that is sometimes hard, if not impossible, to fix. The environment can cause issues in the human pose estimation process if it is not controlled. The environment can be split into multiple categories, which are discussed in the following sections. 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 lighting is more of an issue.

2.1.1 Background

The background of the scene can cause difficulties in the human pose estimation process. Methods using RGB cameras might have difficulties detecting the difference between the foreground and the background. In RGBD-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.

2.1.2 Lighting

While RGB cameras are not affected by too much light, RGBD cameras are heavily influenced by light as most detection methods involve some form of light, be it visible to the human eye or not.

Most RGBD cameras use infrared light to determine the depth of the scene, some use a pattern of infrared light that is projected onto the scene and 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 infrared light emitted by the camera. This can cause the depth of the scene to be incorrect 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, $REPLACE_WE$ do all of the recordings in a room without any natural light or at 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 is very limited and may result in a wrong pose.

2.1.3 Objects

The objects in the scene may cause issues in the human pose estimation process if they either occlude the user 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.

Therefore, it is essential to keep the area of detection as clear as possible to avoid any occlusion or interference with the pose estimation.

2.1.4 Chair

If the exercise is performed in a sitting position the chair might influence the accuracy of HPE. For example, wheelchairs pose a problem in some estimators but a significant part of SilverFit's users are using wheelchairs due to health conditions. Furthermore, bulky chairs that go higher than the head may prevent accurate head detection and silhouette estimation, which is also sometimes used instead of the pose.

In some cases it is not possible to change the chair, either there are no other chairs available, or the person has to sit in a wheelchair. If human pose estimation does not work reliably in these cases alternatives have to be found or the skeleton has to be somehow fixed.

2.2 Camera

In this section, $REPLACE_WE$ discuss the difficulties that can occur due to the camera. SilverFit uses a predefined camera setup, which is the same for every customer. This setup is tried and tested and has been used for many years. However, the camera setup can still cause difficulties during human pose estimation.

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.

2.2.1 Distance

The distance of the camera to the user effects the quality of the pose estimation in multiple ways. Firstly, if the user is too close to the camera, the camera might not get the complete body into frame. The legs and arms might be off 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. There are methods which can detect a human pose from a far distance, however, the further away people get from a camera the more challenging it is. Additionally, as mentioned before, depth cameras operate at different depth ranges. If a user is too far away from the camera it wont be visible to the depth camera. This range varies from camera to camera and also depends on the method of detection.

2.2.2 Angle

When considering angles the main focus lays 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 center. Most human pose estimators are trained without roll in mind. The cameras are usually setup so that any roll is minimised.

Additionally, the pitch may introduce or reduce the occlusion of joints by other joints. It also influences which area is best for human pose estimation. 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.

2.2.3 Resolution

The resolution of the camera, or rather the resolution of the 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 specifically range are important for choosing the right resolution. If a user is far away a higher resolution is needed to detect the pose reliably.

This is also the case for RGBD cameras. Most RGBD cameras have a set range at which they operate. Additionally, the further away from a depth camera you get, the more noisy the depth stream becomes.

2.3 Person

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

2.3.1 Clothes

As mentioned earlier, most RGBD 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.

2.3.2 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.

2.3.3 Exercises

Finally, the most important factor is the exercise that is carried out. In this section, $REPLACE_WE$ define some exercises that are easy to detect as well as some exercises that are difficult to detect. $REPLACE_WE$ also discuss the difficulties that cause the exercises to pose issues for human pose estimation.

These exercises might not be the most realistic, but they represent common issues with pose estimation in a reproducible manner. Furthermore, these exercises are not too difficult to perform, which makes them suitable for testing the pose estimators. The difficulty rating of the exercise might 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 the difficulty. The first letter is an identifier that it is an exercise. The first digit indicates the difficulty of the exercise, while the second digit indicates the number of the exercise. The difficulty is rated from 0 to 4, where 0 is the easiest and 4 is the most difficult. The exercises are divided into four categories: trivial, easy, medium and hard.

Trivial Exercises

Trivial exercises are exercises that are easy to detect and are therefore good for testing the pose estimators. These exercises are not too difficult to detect and are therefore good for testing the pose estimators. The exercises do not involve any movement, which makes detecting the joints easier.

E-0.00 - Arms hanging to the side In the most trivial case, the person is standing still with their arms stretched to the side. In this case, the person is not moving and the joints are not changing position. This is the easiest case for human pose estimation, as the joints are always in the same position. However, this is not a realistic case, as the person is not exercising but it offers a baseline for the other exercises.

E-0.01 - Arms extended to the side Another trivial case is to extend the arms to both sides of the body.

Easy Exercises

Easy exercises are essential for creating a good baseline of how the pose estimators should work. These exercises are not too difficult to detect and are therefore good for testing the pose estimators. The exercises include no self-occlusion and are recorded in a standing position, which is generally the easiest position to detect.

E-1.00 - Raising the arms to the side The first easy exercise is only a small step up from the trivial exercise. In this exercise, the person raises their arms to the side. This exercise is easy to detect, as the arms are raised to the side and the joints are not occluded by the body. Furthermore, the person is standing still, which reduces the possibility of occlusion as well. However, now the arms are moving. This should not pose a problem for the pose estimators, as the arms are not moving too fast. However, it is important to note that the arms are moving, as this can cause issues in some pose estimators.

E-1.01 - Raising the arms to the front A slightly more challenging exercise is when the user raises the arms to the front. This exercise is slightly more challenging than the previous exercise, as the arms are now occluding themselves.

E-1.02 - Raising the arms to the front In a standing position raise first the right knee to the front and then the left knee to the front.

E-1.03 - Raising the arms to the front Finally, in a sedentary position keep both arms hanging to the side. Sitting positions are more challenging to detect than standing positions, as the joints are more occluded by the body. However, this exercise is still easy to detect, as the arms are not moving and the joints are not occluded by the body.

Medium Exercises

Exercises performed in a seated position are harder to detect. Medium exercises focus on exercises, which are performed in a seated position. These exercises only involve arm movement which is easier to detect.

E-2.00 - Raising the arms to the side The first medium exercise is similar to the easy exercise, but now the person is sitting down. Additionally, the user will be holding weights to increase the difficulty of the exercise.

E-2.01 - Raising the arms to the front As with the previous exercise, the user will be sitting down and holding weights. However, now the user will be raising the arms to the front.

E-2.02 - Crossing the arms In the next exercise, the user crosses their arms in front of the body. This exercise is slightly more challenging than the previous exercise, as the arms are now occluding themselves, as well as the upper body.

E-2.03 - Raising the knee The first exercise is when the user raises the knee. This exercise had to be reworked at SilverFit to function well since the pose estimation was too unreliable. From a neutral sitting position with knees at around 90 degrees, the user lifts the knee to a 45-degree angle. Meanwhile, the arms are down to the side.

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 the 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 exercises.

Tölgyessy et al. found that facing away from the camera decreases the accuracy of HPE due to self-occlusion. [32]

E-3.00 - Bowing toward the camera Finally, the user is standing still and bowing forward. Most pose estimators use the head as an anchor point from which the rest of the body is calculated. By bowing forward the head is not perfectly visible and therefore issues are caused.

E-3.01 - Raising the knee leaning forward Additionally to raising the knee, the user will now lean forward, to emulate bad posture. This has the same effect as the previous exercise, as the head is not perfectly visible. However, this exercise is more challenging, as the user is now in a sitting position and the joints are more occluded by the body.

E-3.02 - Raising the knee leaning forward facing away from the camera The user will now lean forward and the body will face away from the camera at a 20-degree angle. This leads to more occlusion and the complete lack of visibility of one of the arms.

Chapter 3

Data Processing

One of the most important aspects of the model development process is the data processing. The data is used to train the model and to evaluate the model. Therefore, it is important to ensure that the data is of high quality. During the development of the project, multiple different approaches were considered. First, $REPLACE_WE$ will discuss the different approaches that were considered and then $REPLACE_WE$ will discuss the approach that was chosen. $REPLACE_WE$ then discuss difficulties that were encountered during the data processing process.

This research proposes a method that enables the fault detection of human pose estimation. Therefore, part of the data processing process is human pose estimation. In section 1.4, $REPLACE_WE$ have already discussed different methods that can be used for human pose estimation. In the scope of this thesis, $REPLACE_WE$ will only focus on a single human pose estimator. This has multiple reasons. Firstly, $REPLACE_WE$ do not intend to develop a general fault detector. Different pose estimators have different flaws so increasing the pool of pose estimators would presumably create a more general fault detector but less accurate fault detector. Some faults might be generally applicable while others are more specific to a specific estimator. This is especially true if a pose estimator uses a different modality, e.g., OpenPose solely uses RGB data, whereas the human pose estimator proposed by C. Zimmermann et.al utilises RGBD data[3, 36]. Furthermore, different pose estimators usually do not result in the same joints as they have been developed for different purposes. For example, OpenPose detects multiple joints in the face, while others use a single head joint. This discrepancy makes it difficult to develop a general fault detector for multiple pose estimators.

We chose the same pose estimator that is used by SilverFit. SilverFit uses Nuitrack which was developed by 3DiVi Inc[1]. Nuitrack does not offer any official white paper or mention how exactly their pose estimation works let alone which modality is used. On further inquiry, a representative notes that since around 2013 Nuitrack uses a similar technique as mentioned by J. Shotton et.al in their paper *Real-time human pose recognition in parts from single depth images*[27]. The method uses random forests to estimate the human pose. However, more recently Nuitrack has switched to Deep Learning with Convoluted Neural Networks, which is becoming more and more the industry standard for computer vision tasks such as human pose estimation.

[1] `https://nuitrack.com/`

3.1 Data acquisition

Different modalities were captured to create a dataset that reproduces a real-world application of human pose estimation for RGBD cameras. The different modalities are RGB data, depth data, and joint data. While the Nuitrack SDK offers to capture the data from the RGBD 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 RGBD+HPE capturing and labelling tool was developed.

FESDData has two main uses which are interlocked. Firstly, it allows capturing predefined, as well as custom, exercises repeatedly automatically making the capturing experience when capturing many different actions more comfortable. Secondly, it allows reviewing and labelling the captured data with error labels. The lightweight nature of FESDData allows it to seamlessly capture both the RGBD stream and the skeleton data at the same time while ensuring a stable fast framerate. The dataset that is used by FESDModel was captured at a framerate of 30 frames per second.

3.2 Data layout

As mentioned earlier, the data is made up of multiple different streams or modalities. There are two separate visual streams, the RGB stream, and the depth stream, as well as the estimated human pose, the time stamps, and the recording metadata.

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. Each joint is stored with real-world 3D coordinates which are stored in meters. These real-world coordinates are labelled x, y, and z. Additionally, the 2D projection and depth of the joint are stored in image coordinates and meters for the depth. The 2D projection is labelled u and v 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 explained in section 3.2.1. Finally, the timestamp of each frame is stored.

3.2.1 Data labeling

A large part of the data preparation is the labelling of the data. The data is labelled with error labels. Two areas can be labelled as erroneous. First, there 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 in the wrong place.

For example, the estimator might label the left foot as the right foot. This is a common error, especially when the limbs are close too 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.

In the data, no error is denoted with 0. If a joint is not detected at all, it is labelled with 1. If a joint is detected in the wrong place that is outside of the body, or somewhere where there should not be a joint, it is labelled with 2. If a joint is detected in the approximate position of where another joint should be then it is labelled with 3. If the whole skeleton is in the wrong position it can be labeled as faulty and subsequently, every joint will be labeled with 2.

Implicitly, this creates 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.

3.3 Dataset

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 the camera is mounted at $175cm$ above the floor. The camera that is used has an accelerometer which was used to adjust the camera angle relative to the ground. The camera is angled downward at a 70° angle.

3.3.1 Problem Sets

Different problem sets are used to create different versions of the model. The problem sets are defined by the number of objects that are considered and are defined as erroneous. There are four different problem sets. The first problem set is the *Joint* problem set. In this problem set, each joint is considered as a single object. The second problem set, the *Limb* problem set, is to consider each limb, i.e. the individual arms, legs, torso, and head. The third problem set is the *Half Body* problem set. In this problem set the upper and the lower body are the areas that are considered. Finally, only the whole body is considered in the *Full Body* problem set.

To determine the threshold at which each area is considered faulty, the number of errors by area was calculated and the 50th percentile was measured. The goal of this is to result in a more balanced dataset. In Figure 3-1 the distribution of joints with errors can be seen. The red line shows the 50th percentile of the distribution, i.e. 50% of the frames contains more errors than the position that is marked. For example, in Figure 3-1, the 50th percentile is at 2 Joints with errors, i.e. if the frame has more than two joints which have an error the Full Body is considered as faulty.

Similarly, Figure 3-2 shows the distribution of joints with errors for the Half Body problem set. The upper half is considered erroneous if one joint in the upper body is faulty. If more than two joints are labelled as incorrect in the lower half of the body then the lower body is labelled likewise.

Finally, the individual limbs are investigated. Figure 3-3, shows the distribution of joints with errors for each of the limbs. All limbs except for the legs are considered faulty if one or more joints in that limb are faulty. The legs are considered erroneous if there is more than one faulty joint.

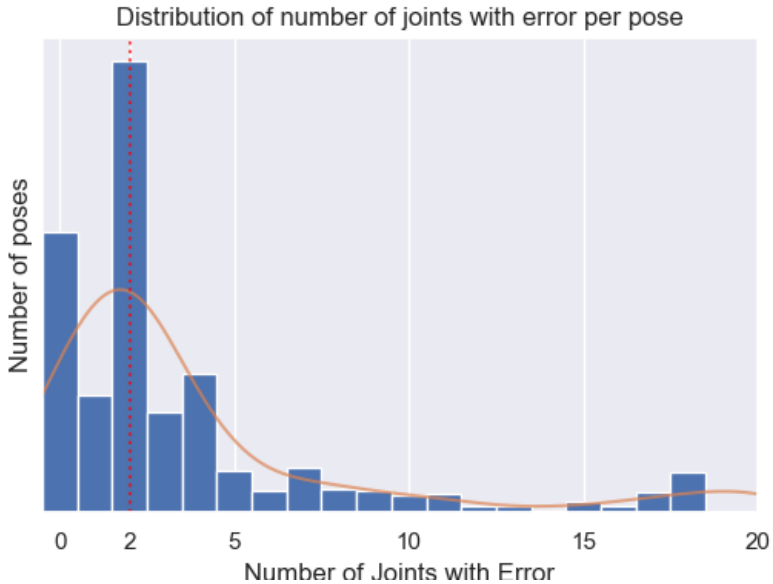


Figure 3-1: The distribution of the number of joints with errors in each frame.

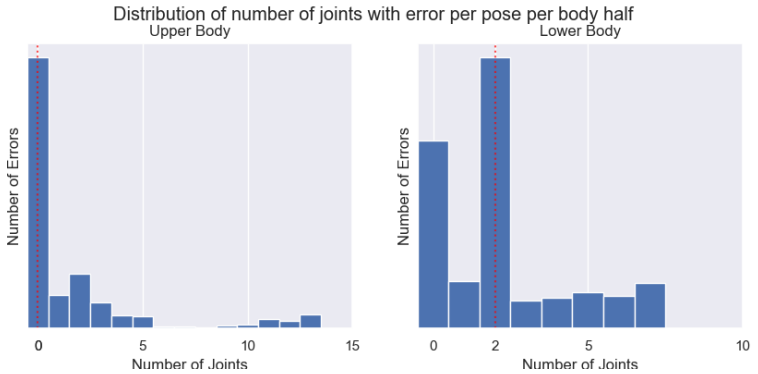


Figure 3-2: The distribution of the number of joints with errors in each frame per body half.

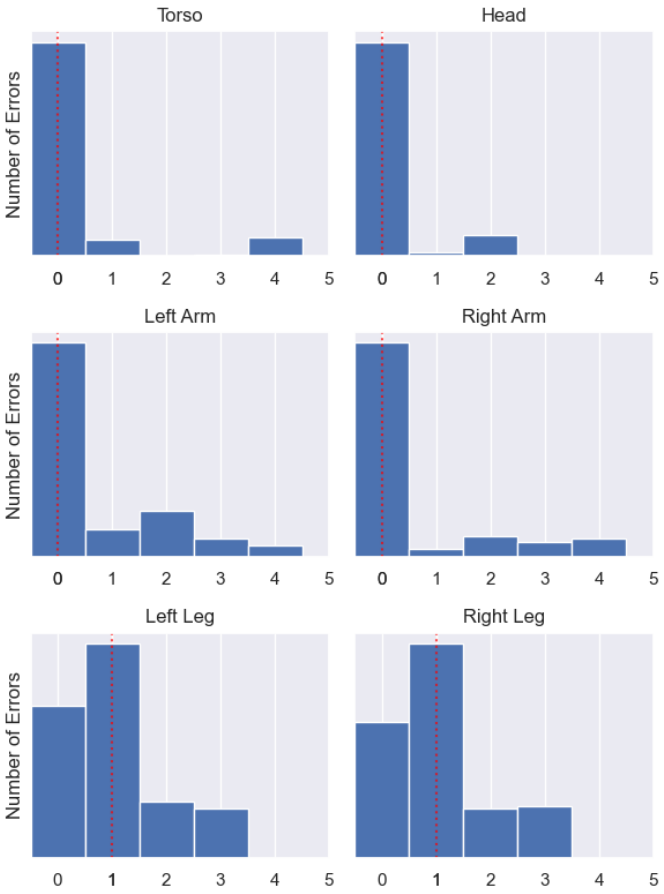


Figure 3-3: The distribution of the number of joints with errors in each frame per limb.

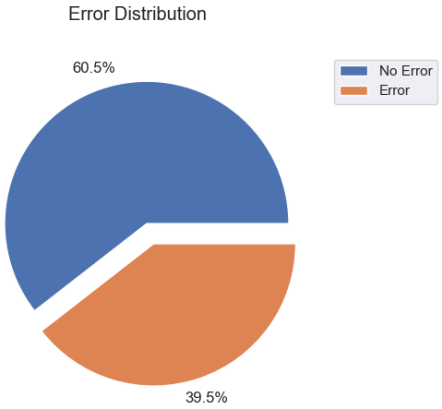


Figure 3-4: The distribution of Errors of the full body problem set.

3.3.2 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 2.3.3 were recorded twice. Each recording session consists of exactly 300 frames.

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 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 3-4 the error distribution of the Full body problem set can be seen. It can be observed that the distribution of errors is reasonably equal with a difference of 10%.

Figure 3-5 gives an overview of the error distribution by the difficulty of the exercise. The figure indicates that the difficulty of the exercise directly influences the percentage of errors that occur.

Half Body Error Distribution

Figure 3-6 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.

The distribution of errors by difficulty can be seen in Figure 3-7. Easy exercises seem to be less error-prone when grouping the joints into body halves. This might be caused by the error threshold that was set before.

Limb Error Distribution

The error distribution of each limb is shown in Figure 3-8. It can be observed that the errors of the limbs are individually very unbalanced. The left arm is the most error-prone

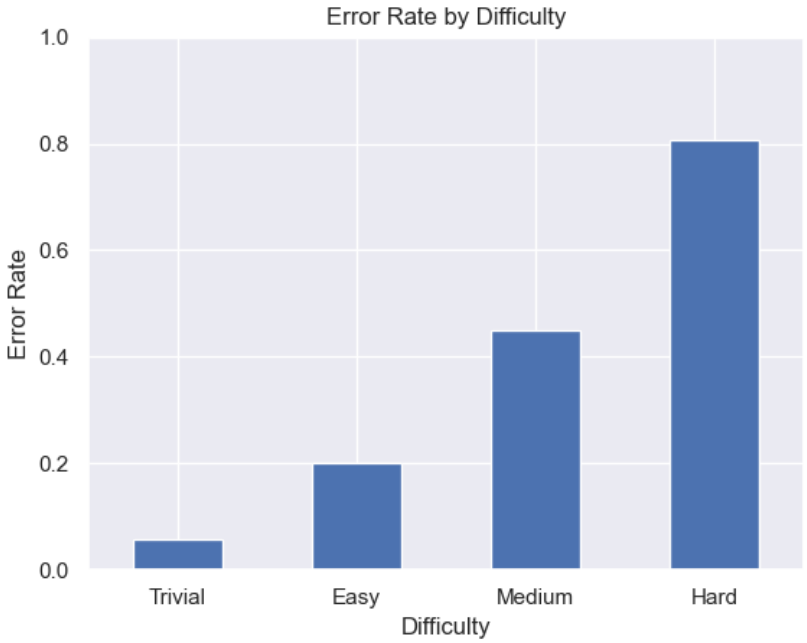


Figure 3-5: The distribution of Errors of the full body problem set by difficulty.

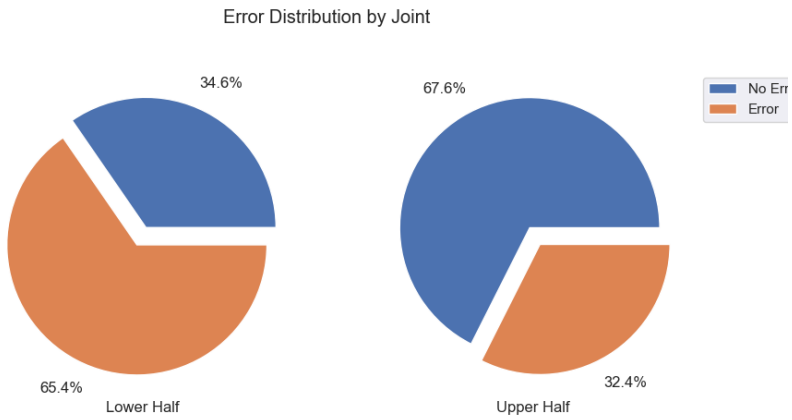


Figure 3-6: The distribution of Errors of the half body problem set.

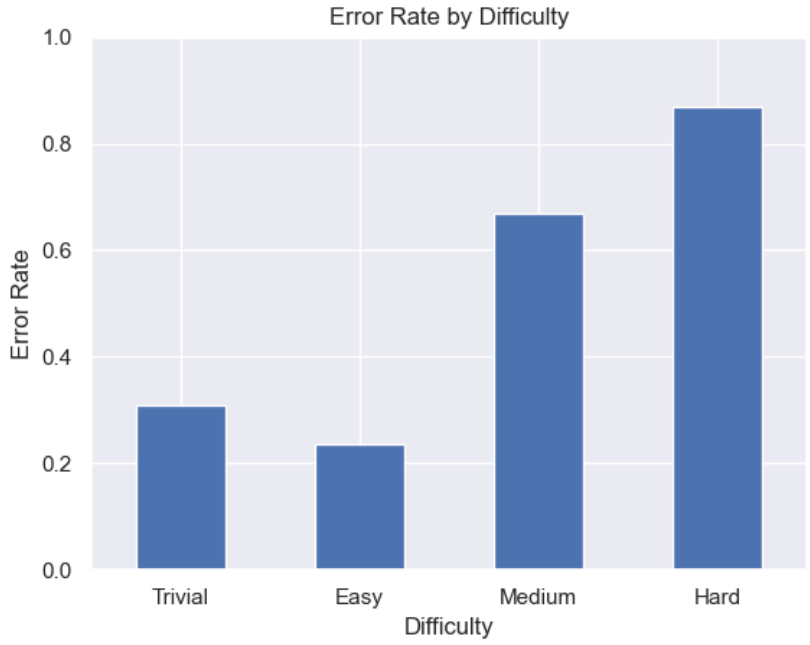


Figure 3-7: The distribution of Errors of the half body problem set by difficulty.

limb with 24.1% of the joints being faulty. The torso is the least error-prone limb with 10.3% of the joints being faulty. This again underlines the assumption that the upper body is more stable than the lower body.

The distribution of errors by difficulty can be seen in Figure 3-9. A less distinct distribution of errors can be seen when grouping the joints into limbs. Only a very small amount of errors occur during trivial exercises. This might be caused by the error threshold that was set before. The majority of errors occurring during trivial exercises are caused by a single faulty joint in the ankle. In Figure 3-3 it is shown that the legs have an error threshold of two faulty joints. This means that the legs are considered faulty if there are more than two faulty joints. The ankle is the only joint that is faulty in trivial exercises. This means that the legs are not considered faulty.

Joint Error Distribution

Figure 3-10 shows that the major part of the errors that occur are Errors with the label 2, i.e. the joint is detected in 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.

In medium exercises, the majority of errors that occur, occur due to missing joints, rather than joints which are in the wrong position. This is shown in Figure 3-11.

A clear distinction between error-prone and stable joints can be seen in Figure 3-12. While in the majority of cases (72.3%) the right ankle is faulty only 4.9% of the left shoulder is faulty. Furthermore, the left and right ankles have a significant number of missing joints, contrary to the other joints, which have a more balanced distribution of error classes. This might be caused by occlusion or a cutoff from the image or a confidence which is too low and therefore discarded by Nuitrack.

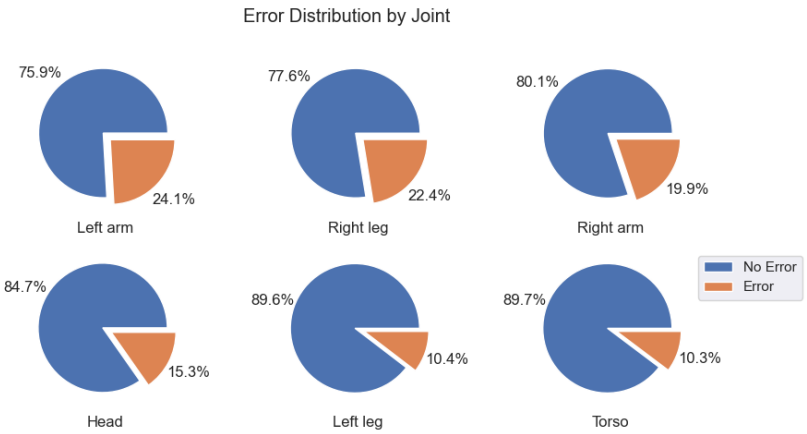


Figure 3-8: The distribution of Errors of the limb problem set.

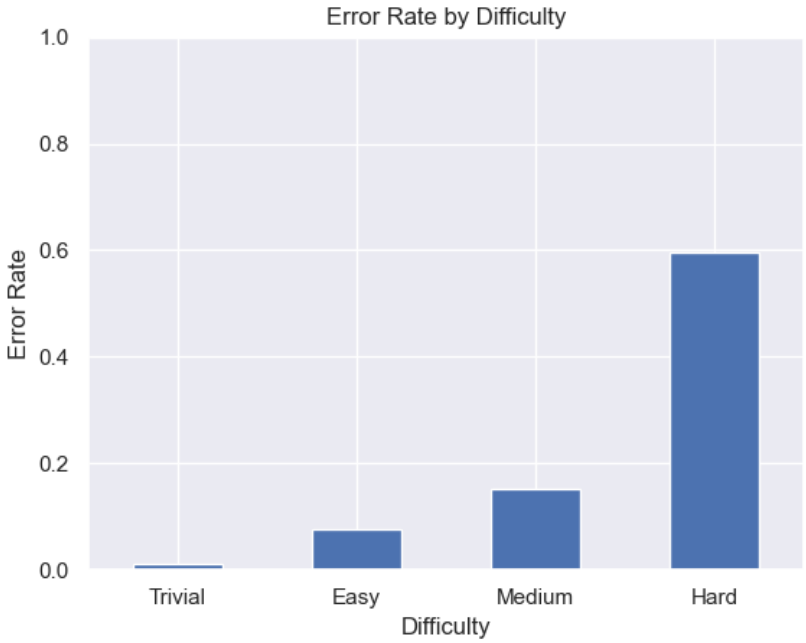


Figure 3-9: The distribution of Errors of the half-body problem set by difficulty.

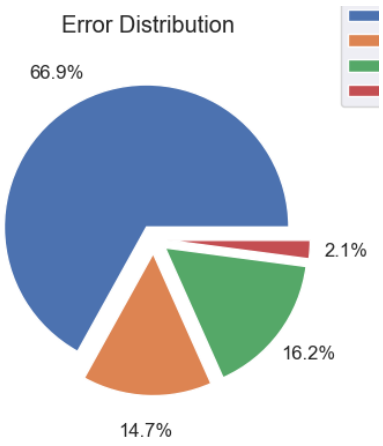


Figure 3-10: The distribution of each error class.

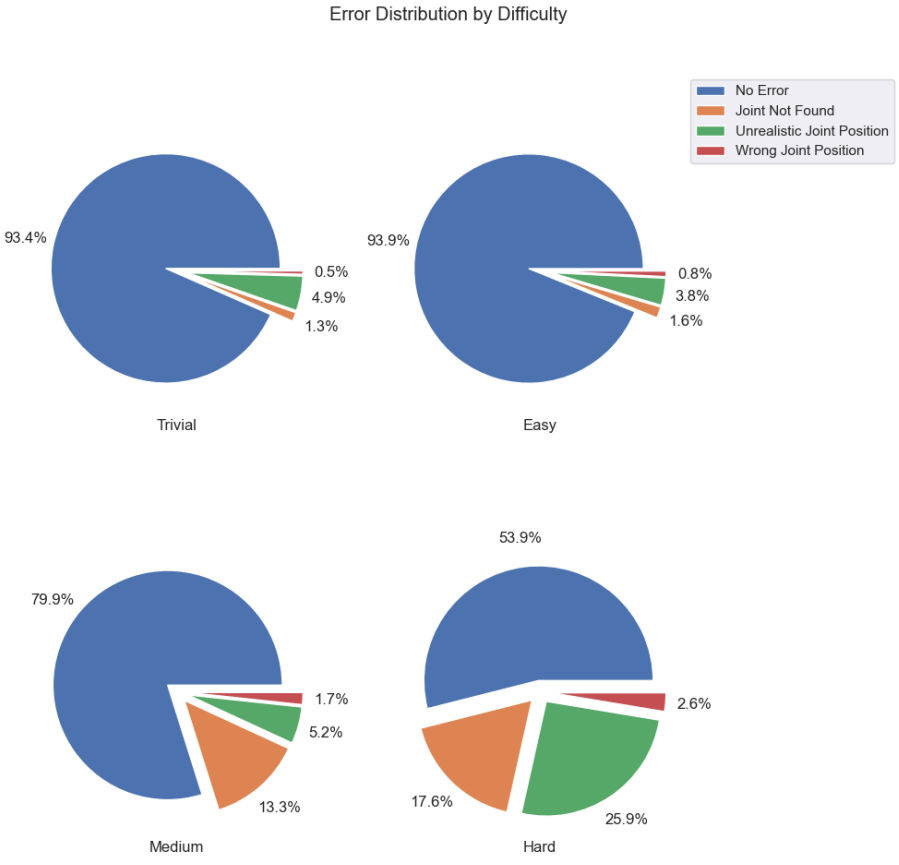


Figure 3-11: The distribution of each error class grouped by difficulty.

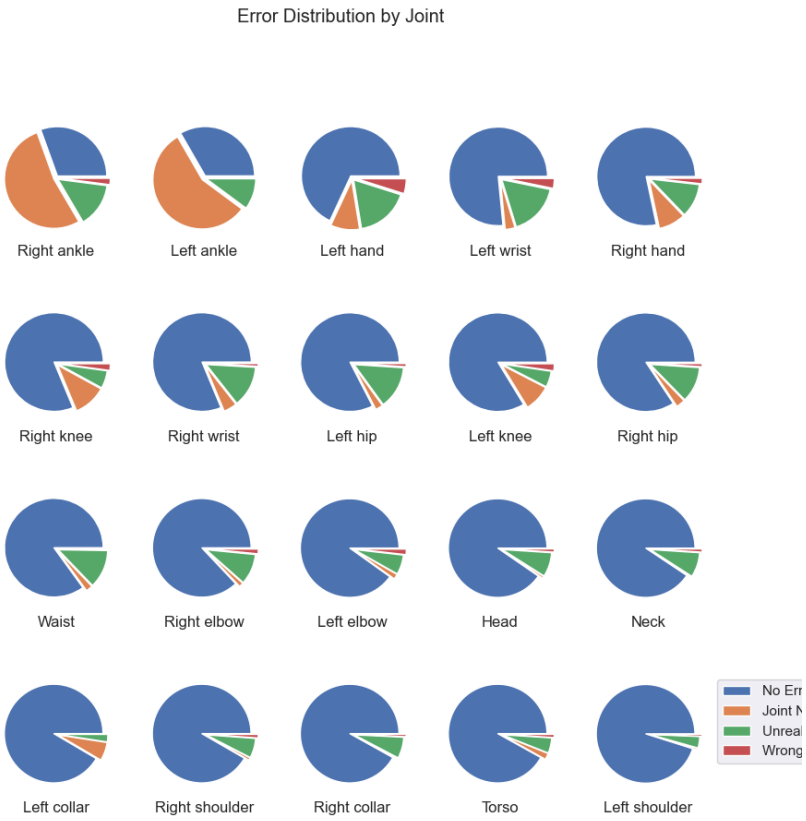


Figure 3-12: The distribution of each error class grouped by joint.

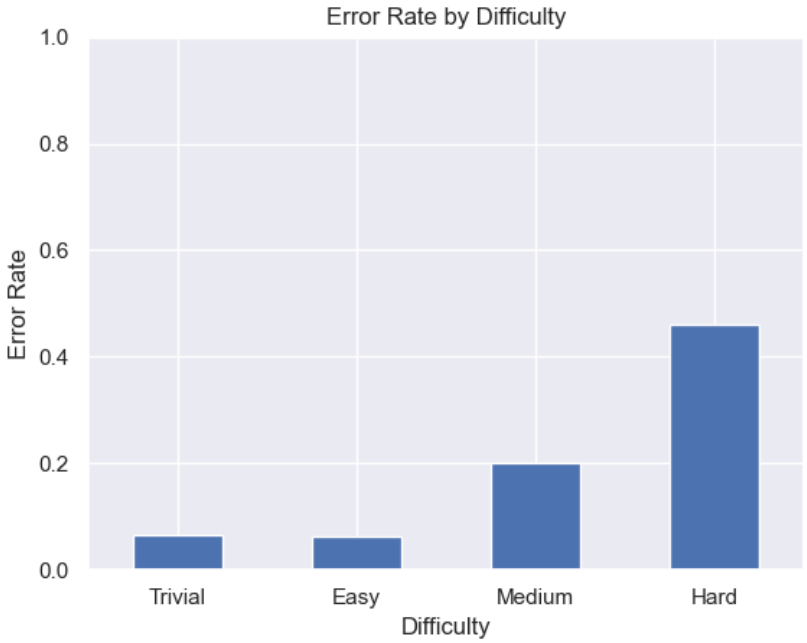


Figure 3-13: The distribution of Errors of the joint problem set by difficulty.

Chapter 4

Model development

While there could be multiple approaches to fault estimation, a deep learning approach has been chosen. The reason for this is that deep learning has shown to be very successful in many different fields, such as image classification, object detection, and image segmentation. The reason for this is that deep learning can learn the features of the data by itself, without the need for manual feature extraction.

Other possible solutions could be to use rule-based systems, which use inverse kinematics, or use frame-to-frame joint comparison to detect discreptancies, however, these are quite limited and might result in either too many false positives or false negatives. Furthermore, these rules, such as the frame-to-frame joint comparison, are not always applicable to all types of movements, and therefore might not be able to detect all types of faults in all cases.

4.1 Model architecture

In the process of this thesis, two versions of the FESD model were implemented. The first approach can be seen in Figure 4-1. In the first approach, the RGB, Depth, and Joint data are passed into three different convolutional neural networks with intermittent max-pooling. The output of the three networks is then concatenated and passed into three fully connected layers. The output of the fully connected layers is a 1D 2, 4, 20, and 80 tensors of multi-object multi-class values depending on the problem set. The different problem sets are discussed in section 3.3.1. However, for this approach, the amount of data that was captured was not sufficient so no satisfying results could be achieved. Therefore, a second approach was implemented.

The second approach is relying on transfer learning, in which a pre-trained model is used as part of a new network. To choose a network, multiple different networks have been compared, which can be seen in Figure 4-2. Since the application is in gaming a lightweight model which does not impact the performance much is preferred. Therefore, the models are compared by the number of number of floating-point operations (FLOPS) to their Accuracy on ImageNet-1K. Table 4.1 shows the top 5 models according to their accuracy and performance.

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%. The model architecture can be seen in Figure 4-3. The model is split into two parts. The first part is the feature extractor, which is the EfficientNet v2 S. The second part is the classifier, which is two fully connected

Table 4.1: The top 5 models according to their accuracy and performance. The models are sorted by their GFLOPS and their Top-5 Accuracy[2]. 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

Layers. The output of the classifier is a 1D 2, 4, 20, or 80 tensors of multi-object multi-class values depending on the problem set. Additionally, the input is merged into a single image so that the EfficientNet v2 can be used to extract the features of all modalities.

4.2 Data preparation

To successfully train FESD 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. The data balancing is done to ensure that the model is not biased toward any particular error label.

The finished data preparation pipeline can be seen in Figure 4-4.

The images that are stored by *FESDData* are inherently the same size. The joints, however, are stored within a JSON file containing the coordinates of each joint in 2D and 3D. To further process the 2D joint data is drawn on an image that has the same dimensions as the RGB and Depth image.

4.2.1 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.

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, at random 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 4-4 where they are applied to a sample frame from the dataset.

4.2.2 Data merging

While there are different ways of applying transfer learning to the problem at hand, $REPLACE_WE$ chose to merge the different modalities into a single tensor. This allows $REPLACE_US$ to

use the EfficientNet v2 as a feature extractor for all modalities, without the added computations of extracting the features of each modality individually. The data is merged by assigning each modality to a channel in an RGB image.

In Figure 4-4 $REPLACE_WE$ can see the different modalities as they are merged into a single tensor. The RGB image is transformed into greyscale and assigned to the red channel, the depth image is scaled to a value between 0 and 255 and assigned to the green channel, and the joint coordinates are assigned to the blue channel. The data is then passed into the EfficientNet v2 as a single RGB image.

4.2.3 Data balancing

In the ordinary case, human pose estimation is not meant to produce faulty results. In the selected exercises it is aimed to produce faulty results. However, this does still not produce a balanced dataset. In section **??**, the statistics of the dataset are shown. Especially, in Figure **??** it can be seen that the dataset is imbalanced. Most notably for the problem set *Half* and *Full* where the error label *No Error* is overrepresented.

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 errors is considered as a single error label. This means that frames that have the same number of erroneous areas are weighed the same.

An example of the distribution of errors before and after balancing can be seen in Figure **??** for each problem set.

4.3 Model training

To train a neural network with supervised learning, you first pass the input data into the network and forward it through the defined layers. Then a loss is calculated an is is back propagated through the network to adapt the weights accordingly.

As mentioned earlier, error or anomaly detection for human pose estimation can be seen as a multi-class multi-object classification problem. The loss function needs to be chosen such that it best reflects the data and the efficacy of the model. In most classification problems *Cross Entropy Loss* is calculated and propagated through the network to adapt the weights and convolutions accordingly. Cross entropy loss calculates the soft max of the result and compares it to the ground truth. The soft max calculates the probability of each index in a list based on the value at that index. Cross Entropy Loss penalises the results based on the probability of the target class.

However, since multiple objects, or areas, are considered the loss function has to be split the loss function and apply it area wise. This is done by calculating the loss for each area and then summing the losses and calculating the average. This is done to ensure that the model is not biased toward any particular area.

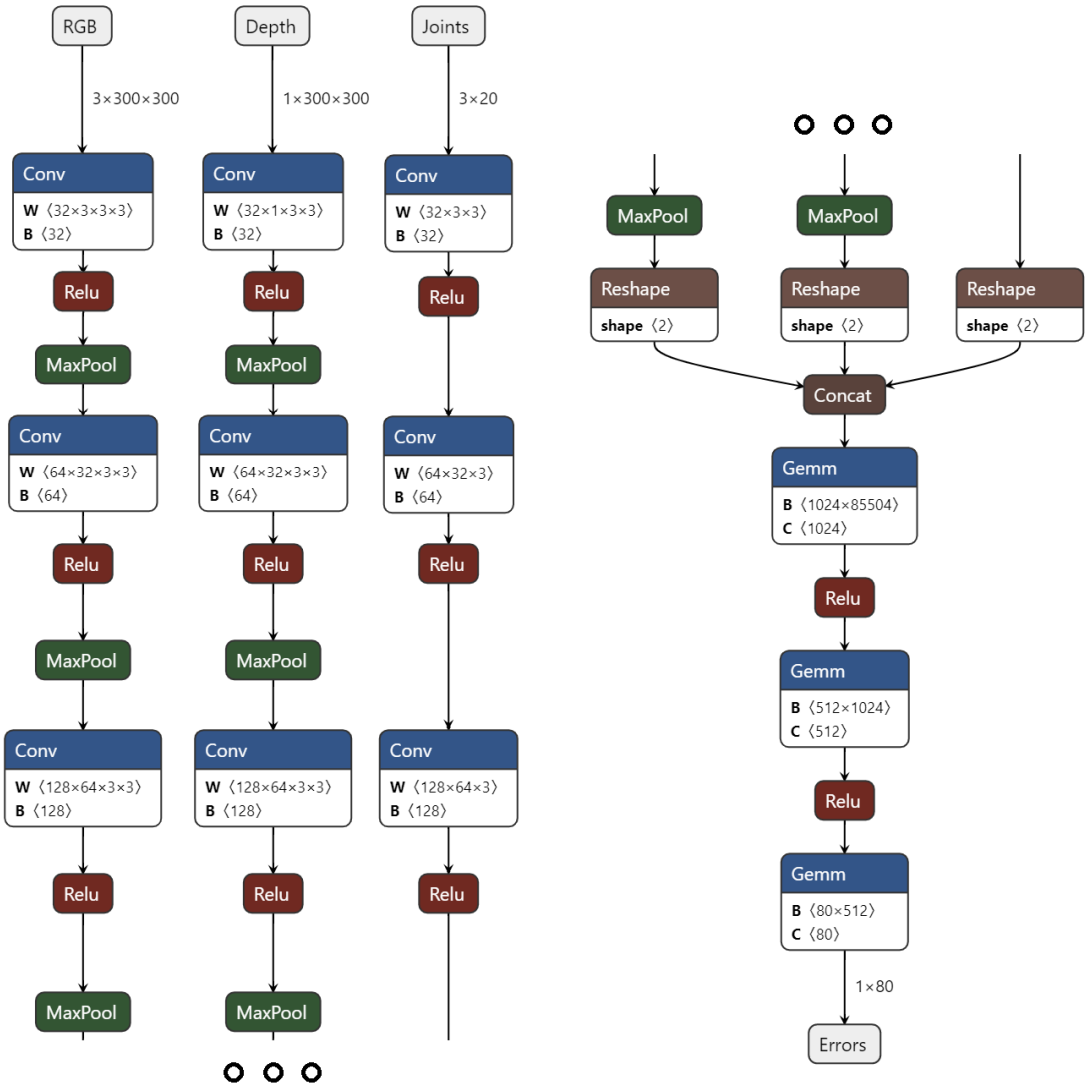


Figure 4-1: Original FESDModel architecture with three different inputs; RGB, Depth and Joint data. After three convolutions the three streams are concatenated to be passed into three fully connected layers. The output is a 1D 80 tensor of multi-object multi-class values.

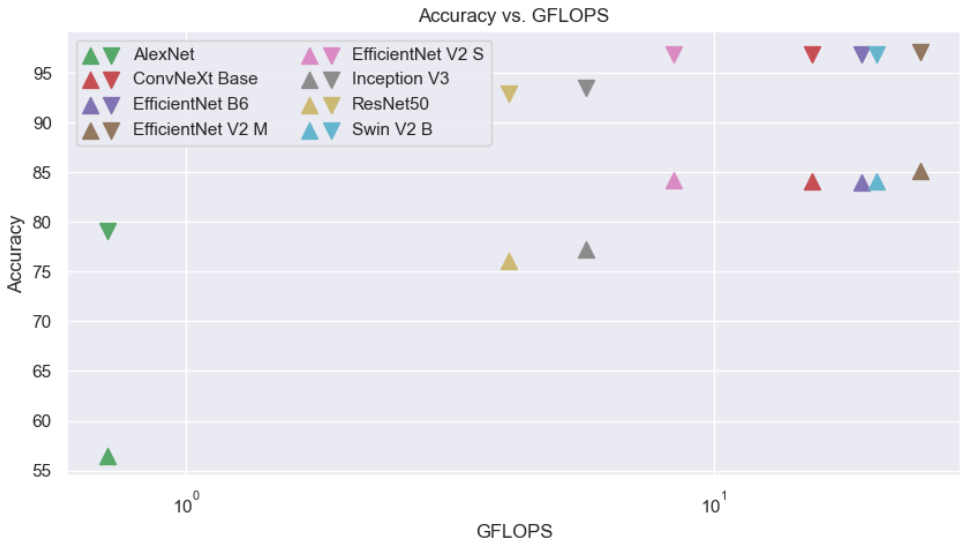


Figure 4-2: The comparison of different networks by their GFLOPS and their Top-5 Accuracy. The models are sorted by their GFLOPS and their Top-5 Accuracy[1]. The models are EfficientNet V2 S, ConvNeXt Base, EfficientNet B6, Swin V2 B, and EfficientNet V2 M. Additionally, AdamNet, ResNet-50 and Inception-v3 are added as a reference.

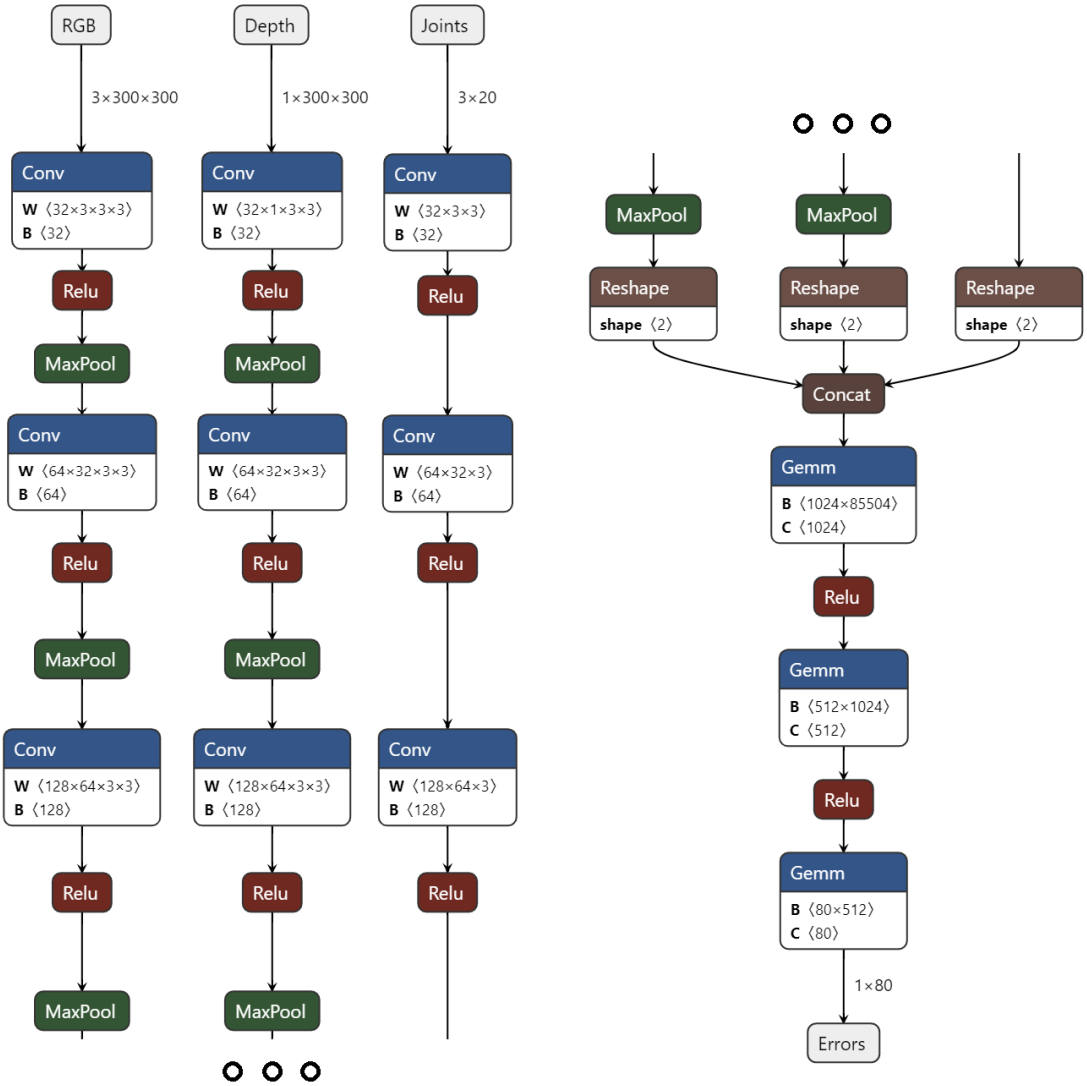


Figure 4-3: FESDModelv2 architecture with transfer learning. TODO

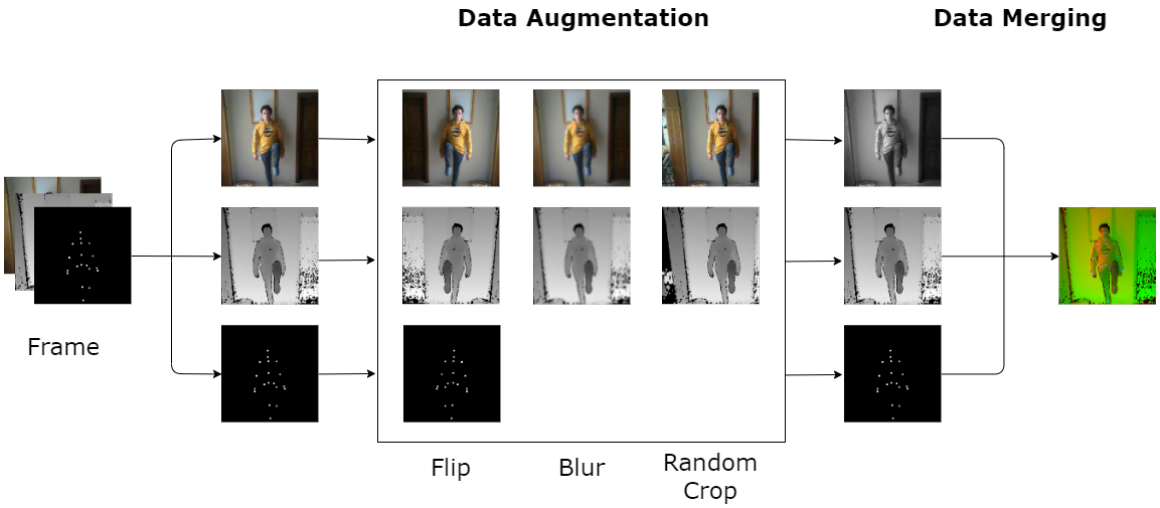


Figure 4-4: The data preparation pipeline. The data is randomly flipped, Blurred, and Cropped and then merged into a single RGB image.

Chapter 5

Results

In this chapter, the results of the experiments are presented. The results for each different problem set is presented seperatly.

5.1 Full Body Error Estimation

??

The first and most general model that was developed was the full body model. The full body model was trained to give a single error label as an output given an image as an input. The results of the training process of the Full Body model can be seen in Figure 5-1.

The result of the training is rather disappointing. The testing showed that the model only performs well on the training data. The model does not generalise well to new data. The reason for this is that the model is overfitting. The model is overfitting because the model is too complex for the amount of data that is available. The model has 21.5 million trainable parameters. This is too many parameters for the amount of data that is available. The model is not able to learn the underlying patterns of the data, but rather memorises the training data. This is also shown in the training and testing loss. The training loss is decreasing, but the testing loss is increasing. This is a clear sign of overfitting.

5.1.1 Confusion Matrix

The confusion matrix of the full body model is shown in Figure 5-2. It can be seen that the model only predicts the error label 0 - No Error. This is because the model is overfitting.

Figure 5-1: The training results of the full body model.

Figure 5-2: The confusion matrix of the full body error estimation model.

5.2 Half Body Error Estimation

The second model that was developed was the half body model. The half body model was trained to give two error labels as an output given an image as an input, one for the lower body and one for the upper body. The results of the training process of the Half Body model can be seen in Figure 5-3.

Confusion Matrix

The confusion matrix of the half body model is shown in Figure 5-4. It can be seen that contrary to the full body model the half body model variies the error label.

5.2.1 Emulated Full Body

The results of the model can be used to emulate the full body model. Therefore, the two models can be compared. The results of the model are shown in Figure 5-5. It can be seen that the half body model outperforms the full body model. The reason for this is that the full body model estimates purely positive results, whereas the half body models variies as is shown in the $\frac{p}{p+n}$ graph, which shows the percentage of positive guesses.

5.3 Limb Error Estimation

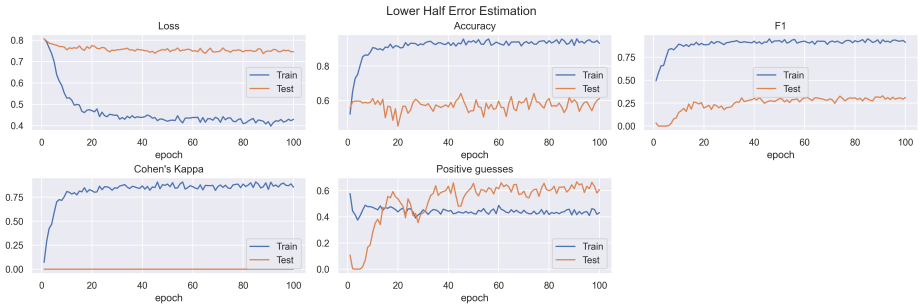
The results for the limb error estimation are the same as for the full body error estimation. The model is overfitting and therefore only predicts the error label 0 - No Error for most cases. The results of the training process of the Limb model can be seen in Figure 5-7.

5.3.1 Confusion Matrix

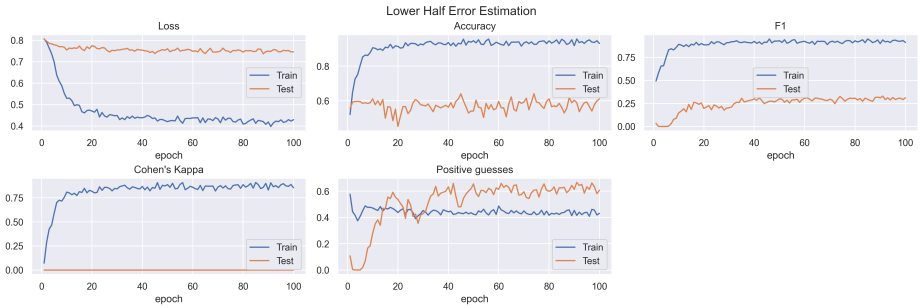
The confusion matrix of the limb model is shown in Figure 5-8. It can be seen that the model only predicts the error label "No Error" or "Error" for most joints without variation. This is because the model is overfitting.

5.3.2 Emulated Full Body

As with the half body Error estimator, the full body is emulated with the result of the limb error estimation. The results can be seen in Figure 5-9.



(a) Upper Body Error Estimation



(b) Lower Body Error Estimation

Figure 5-3: The training results of the half body error estimation model.

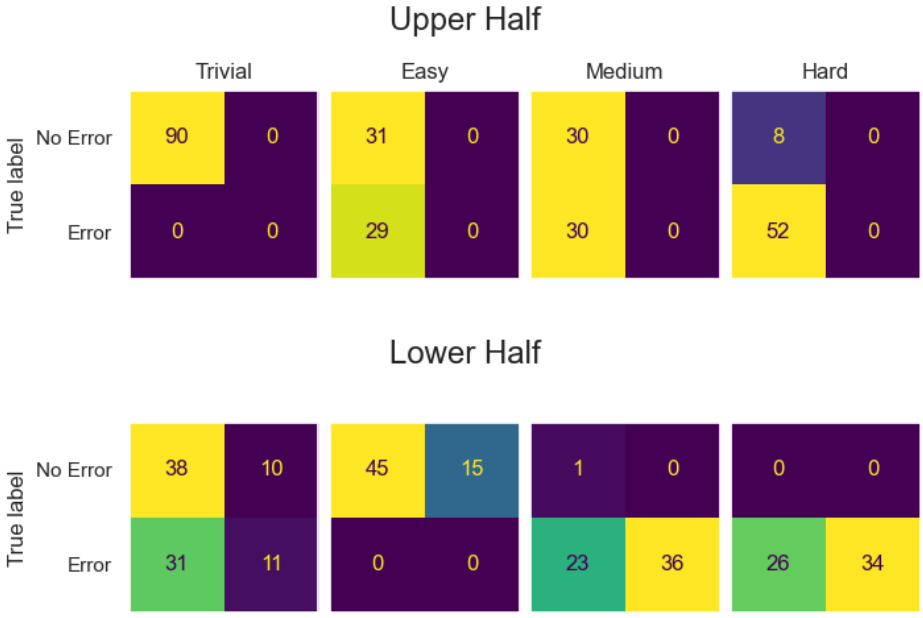
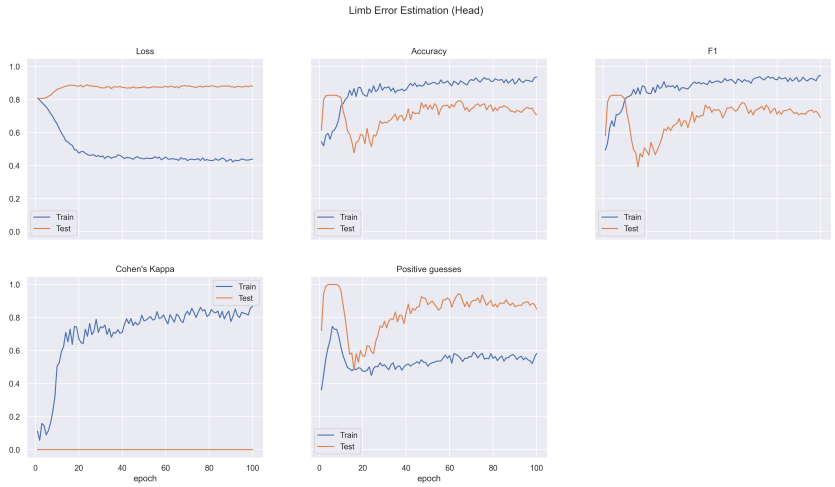
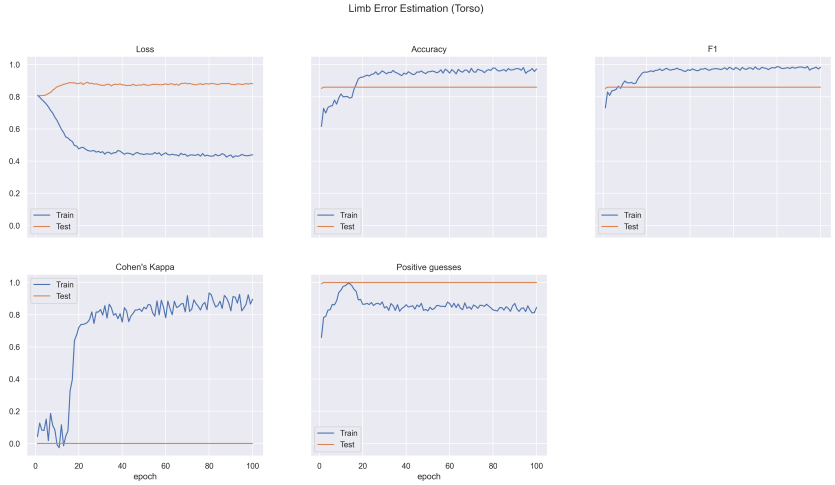


Figure 5-4: The confusion matrix of the half body model.

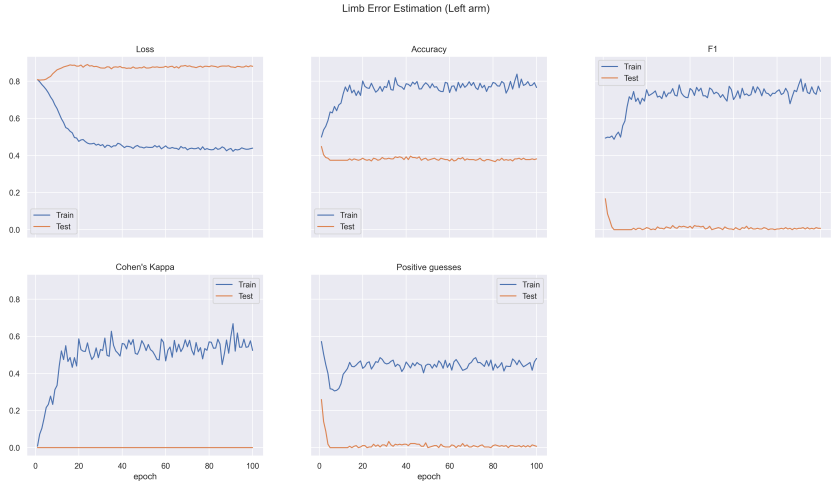
Figure 5-5: The results of the half body model when emulating the full body results.



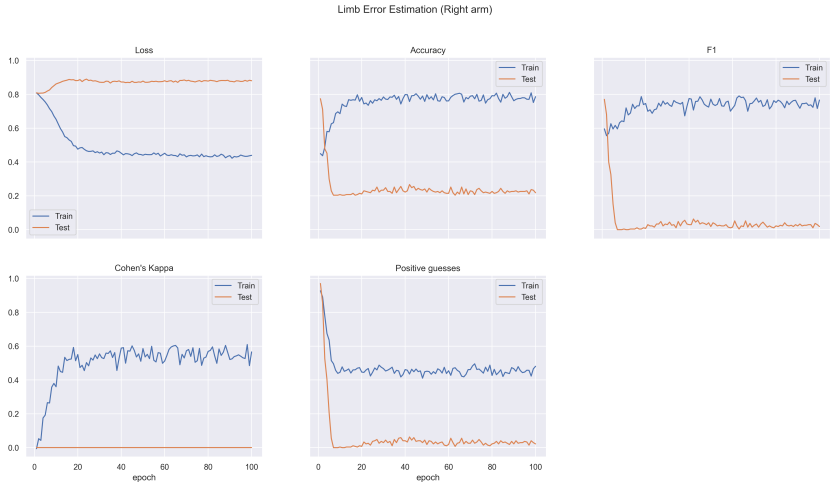
(a) Head Error Estimation



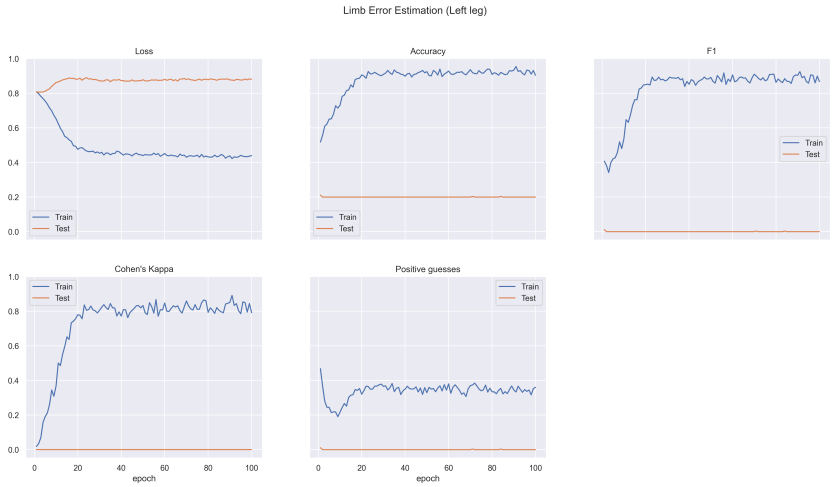
(b) Head Error Estimation



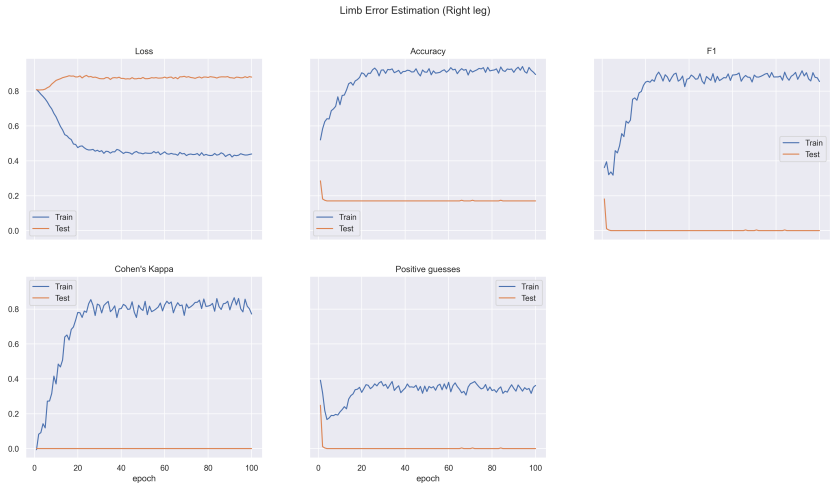
(c) Left Arm Error Estimation



(a) Right Arm Error Estimation



(b) Left leg Error Estimation



(c) Right leg Error Estimation

Figure 5-7: The training results of the limb error estimation model.

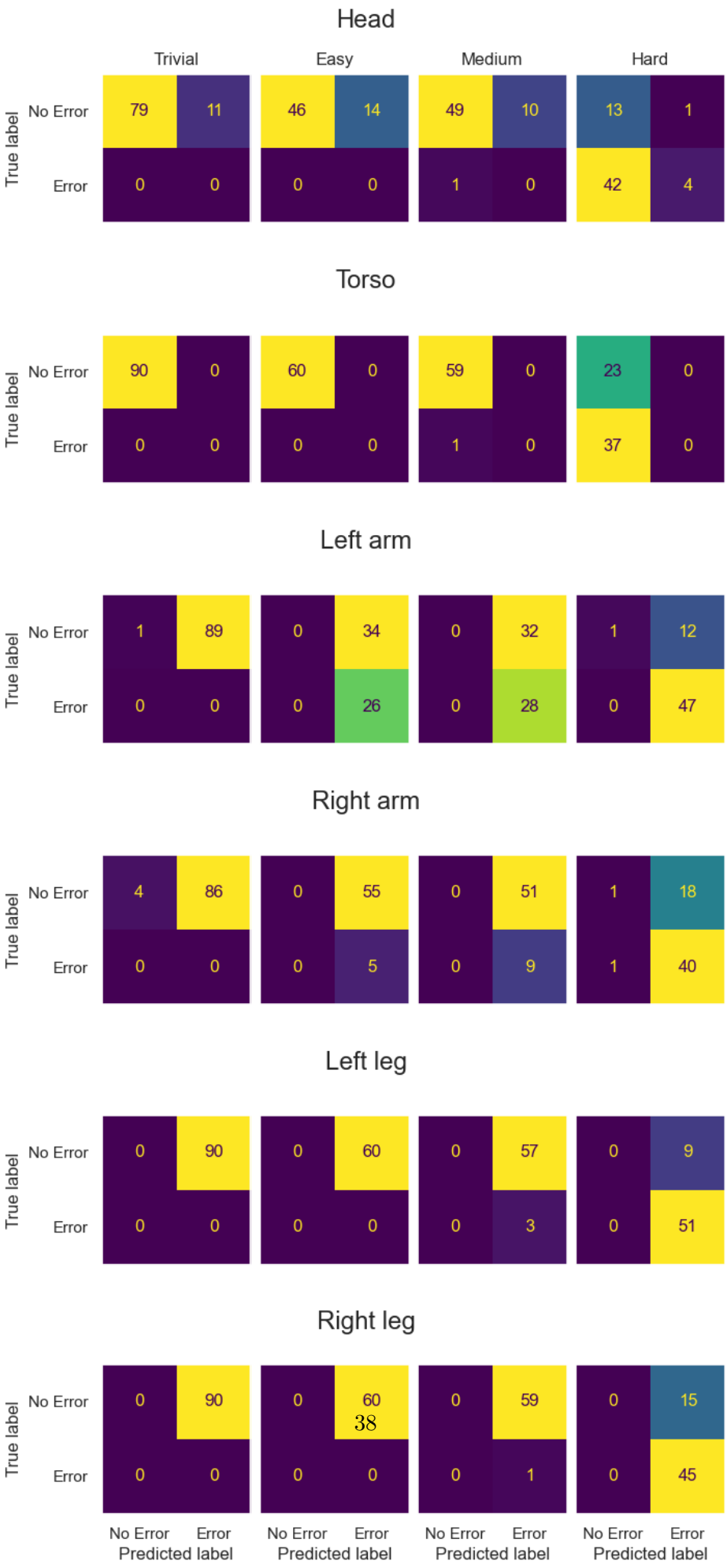


Figure 5-8: The confusion matrix of the limb model.

Figure 5-9: The results of the Limb model when emulating the full body results.

Figure 5-10: The results of the Limb model when emulating the half body results.

5.3.3 Emulated Half Body

Additionally, the half body is emulated with the result of the limb error estimation. The Head, Torso, and Left and Right arm are considered as the upper body, and the Left and Right leg are considered as the lower body. The results of the model are shown in Figure **??**.

5.4 Joint Error Estimation

The results for the joint error estimation are the same as for the full-body error estimation. The model is overfitting and therefore only predicts the error label 0 - No Error for most joints. The results of the training process of the Joint model can be seen in Figure 5-11.

5.4.1 Confusion Matrix

The confusion matrix of the joint model is shown in Figure 5-16(TODO: split that graph, its huge). It can be seen that the model only predicts the error label 0 - No Error for most joints. This is because the model is overfitting.

5.4.2 Emulated Full Body

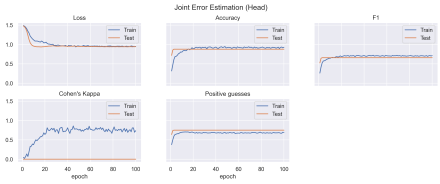
As with the half-body Error estimator, the full body is emulated with the result of the limb error estimation. The results can be seen in Figure 5-18.

5.4.3 Emulated Half Body

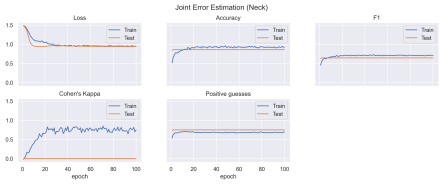
Additionally, the half body is emulated with the result of the limb error estimation. The Head, Torso, and Left and Right arms are considered the upper body, and the Left and Right legs are considered the lower body. The results of the model are shown in Figure 5-19.

5.4.4 Emulated Limbs

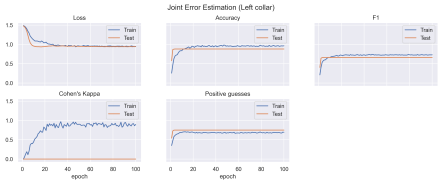
Finally, the results of the joint model are used to emulate the limb model. The results can be seen in Figure 5-19.



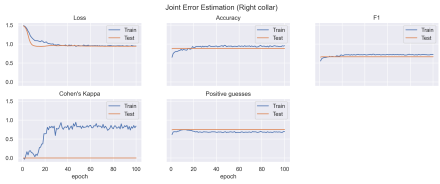
(a) Head Error Estimation



(b) Neck Error Estimation

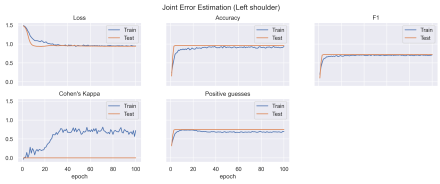


(c) Left Collar Error Estimation

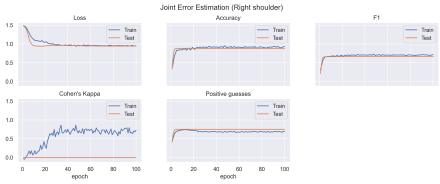


(d) Right Collar Error Estimation

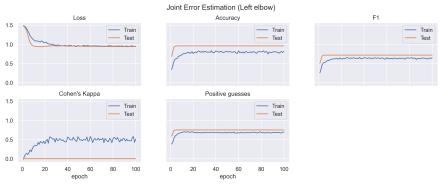
Figure 5-11: The training results of the Joint error estimation model.



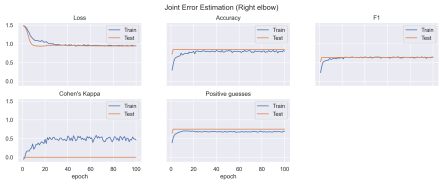
(a) Left Shoulder Error Estimation



(b) Right Shoulder Error Estimation

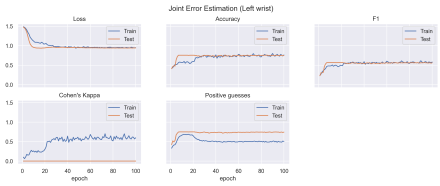


(c) Left Elbow Error Estimation

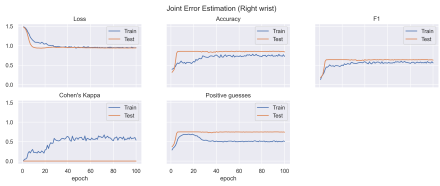


(d) Right Elbow Error Estimation

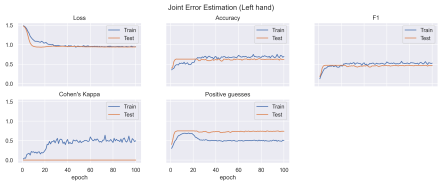
Figure 5-12: The training results of the Joint error estimation model. (cont. 1)



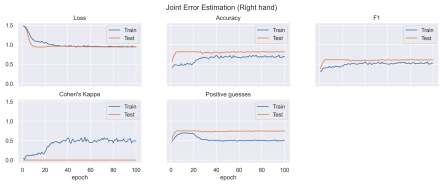
(a) Left Wrist Error Estimation



(b) Right Wrist Error Estimation

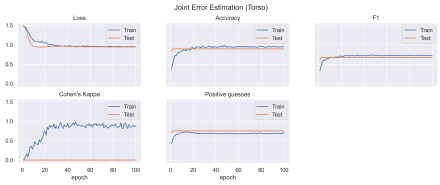


(c) Left Hand Error Estimation

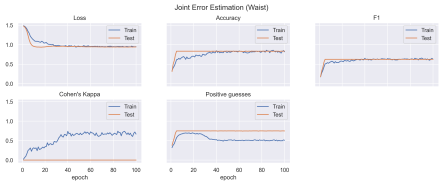


(d) Left Arm Error Estimation

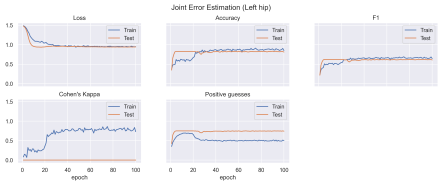
Figure 5-13: The training results of the Joint error estimation model. (cont. 2)



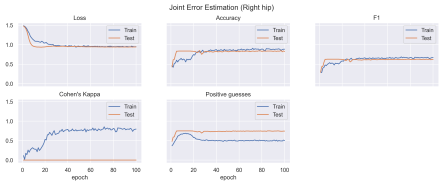
(a) Torso Error Estimation



(b) Waist Error Estimation

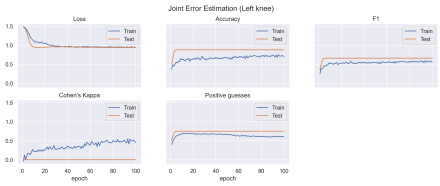


(c) Left Hip Error Estimation

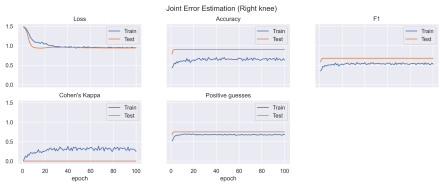


(d) Right hip Error Estimation

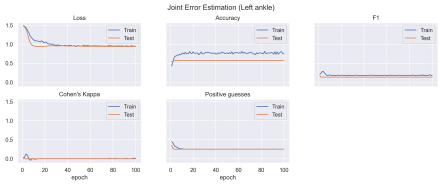
Figure 5-14: The training results of the Joint error estimation model. (cont. 3)



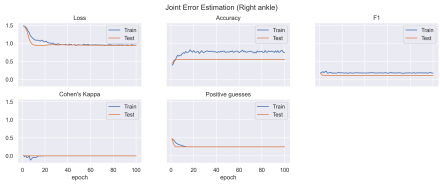
(a) left Knee Error Estimation



(b) Right Knee Error Estimation



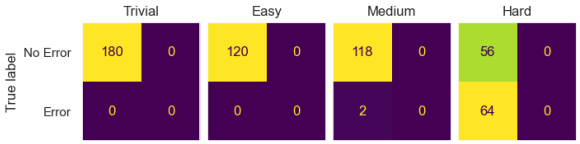
(c) Left Ankle Error Estimation



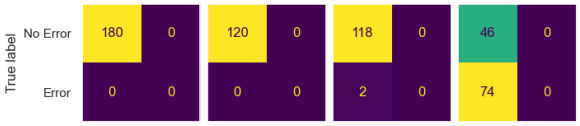
(d) Right Ankle Error Estimation

Figure 5-15: The training results of the Joint error estimation model. (cont. 4)

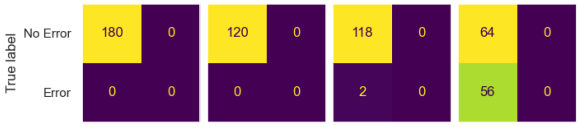
Head



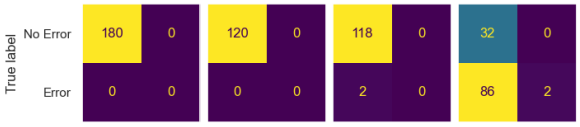
Neck



Torso

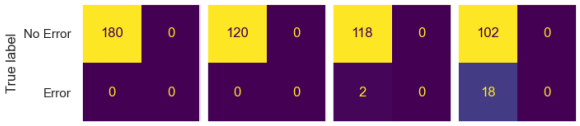


Waist

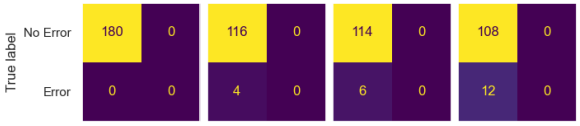


Left collar

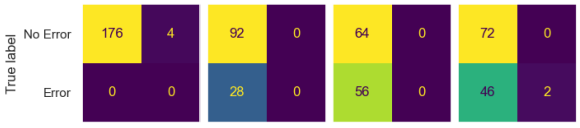
Left shoulder



Left elbow



Left wrist



Left hand

Right collar

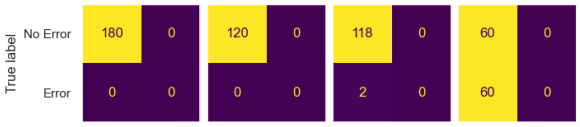


Figure 5-17: The results of the Joint model when emulating the full body results.

Figure 5-18: The results of the Joint model when emulating the half body results.

Figure 5-19: The results of the Limb model when emulating the limb results.

Chapter 6

Conclusion

In Conclusion, $REPLACE_WE$ can see that ... **Not sure what $REPLACE_WE$ can see yet**

6.1 Contribution

In the scope of this thesis, $REPLACE_WE$ developed FESDData and FESDModel, or Fault estimation for Skeleton detection for data collection and model creation. FESDData is the tool that allows $REPLACE_US$ to record, analyse and populate it with pose data using Nuitrack.

FESDData is a tool that is designed to be easy to use and that can be used by anyone, and without much need for setup or tweaking. It allows for the rapid capture of RGBD datasets with pre-labeling of multiple different recordings. The parameters can be adapted to capture datasets for many different application, e.g. Action detection.

FESDModel is the model which $REPLACE_WE$ developed in the scope of this thesis. It utilises the data recorded using FESDData.

The code of this thesis is available on GitHub[1]. The repository is divided into two major parts, FESDData, which contains the C++ implementation of the dataset recorded, FESDModel, which contains the implementation of the model, as well as the Jupyter notebook that was used to train the model. Additionally there is this thesis and all related figures.

6.1.1 Developed Software

UNSURE *Should I write about the software, explain the OpenGL implementation, the ImGui GUI and so on?* ***TODO*** *Change Screenshots to light mode to be consistent with the rest of the thesis (can wait until screenshots are final)*

6.1.2 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 Human Pose estimation. In its simplest application it may be used to warn users of possible errors when the skeleton is not detected correctly. In more advanced cases the information provided by the model could be used to attempt to fix joints through joint position interpolation and

[1] `https://github.com/LeonardoPohl/FESD`

prediction rather than using the faulty joint. Moreover, multiple human pose estimators could be considered resulting in an overall more robust human pose estimation.

Furthermore, if the model proves to have a 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. in [5].

The dataset and the dataset recorder may also be used to further the model development for fault estimation and it can also find application in other areas. The dataset in and of itself can be used for action detection. The exercises are predefined and can be recorded and automatically labeled by FESDData, thereby making it fairly simple to record large amounts of data without requiring a large amount of human work.

6.2 Future work

To improve the quality of FESDModel, more data needs to be collected and labelled in different settings.

figures/FESD/all.png

Figure 6-1: A screenshot of the FESDData GUI streaming a point cloud. The GUI is used to record, playback, visualise and label the data streams from RGBD cameras. The FESDData Gui is written in C++ using the ImGui framework. The point cloud is visualised using OpenGL and a glsl shaders.

Bibliography

[1] Sizhe An, Yin Li, and Umit Ogras. mri: Multi-modal 3d human pose estimation dataset using mmwave, rgb-d, and inertial sensors, 2022.

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[3] 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.

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[5] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback, 2015.

[6] Cristian Sminchisescu Catalin Ionescu, Fuxin Li. Latent structured models for human pose estimation. In *International Conference on Computer Vision*, 2011.

[7] Kenny Chen, Paolo Gabriel, Abdulwahab Alasfour, Chenghao Gong, Werner K. Doyle, Orrin Devinsky, Daniel Friedman, Patricia Dugan, Lucia Melloni, Thomas Thesen, David Gonda, Shifteh Sattar, Sonya Wang, and Vikash Gilja. Patient-specific pose estimation in clinical environments. *IEEE Journal of Translational Engineering in Health and Medicine*, 6:1–11, 2018.

[8] Qian Chen, Ze Liu, Yi Zhang, Keren Fu, Qijun Zhao, and Hongwei Du. Rgb-d salient object detection via 3d convolutional neural networks. 2021.

[9] Yucheng Chen, Yingli Tian, and Mingyi He. Monocular human pose estimation: A survey of deep learning-based methods. *Computer Vision and Image Understanding*, 192:102897, mar 2020.

[10] Shradha Dubey and Manish Dixit. A comprehensive survey on human pose estimation approaches. *Multimedia Systems*, 29, 08 2022.

[11] Abdessamad Elboushaki, Rachida Hannane, Karim Afdel, and Lahcen Koutti. Multid-cnn: A multi-dimensional feature learning approach based on deep convolutional networks for gesture recognition in rgb-d image sequences. *Expert systems with applications*, 139:112829, 2020.

[12] Martin Fisch and Ronald Clark. Orientation keypoints for 6d human pose estimation, 2020.

[13] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988.

[14] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real time motion capture using a single time-of-flight camera. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 755–762, 2010.

[15] Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real-time human pose tracking from range data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

[16] Intel. Tensorflow with intel realsense camera. `https://dev.intelrealsense.com/docs/tensorflow-with-intel-realsense-cameras`. Accessed: 18-02-2023.

[17] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[19] Fabian Kröger. *Automated Driving in Its Social, Historical and Cultural Contexts*, pages 41–68. 05 2016.

[20] Laxman Kumarapu and Prerana Mukherjee. Animepose: Multi-person 3d pose estimation and animation, 2020.

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[22] Yunheng Liu. Contour model and robust segmentation based human pose estimation in images and videos. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8:1–10, 03 2015.

[23] David Pascual-Hernández, Nuria Oyaga de Frutos, Inmaculada Mora-Jiménez, and José María Cañas-Plaza. Efficient 3d human pose estimation from rgbd sensors. *Displays*, 74:102225, 2022.

[24] Bernhard Preim and Monique Meuschke. A survey of medical animations. *Computers and Graphics*, 107, 09 2022.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[26] Bassem Seddik, Sami Gazzah, and Najoua Essoukri Ben Amara. Human-action recognition using a multi-layered fusion scheme of kinect modalities. *IET Computer Vision*, 11(7):530–540, 2017.

[27] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR 2011*, pages 1297–1304, 2011.

[28] Leonid Sigal, Alexandru Balan, and Michael Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87:4–27, 03 2010.

[29] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[31] Liangchen Song, Gang Yu, Junsong Yuan, and Zicheng Liu. Human pose estimation and its application to action recognition: A survey. *Journal of Visual Communication and Image Representation*, 76:103055, 04 2021.

[32] Michal Tölgyessy, Martin Dekan, and Lubos Chovanec. Skeleton tracking accuracy and precision evaluation of kinect v1, kinect v2, and the azure kinect. *Applied Sciences*, 11:5756, 06 2021.

[33] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[34] Tianxu Xu, Dong An, Yuetong Jia, and Yang Yue. A review: Point cloud-based 3d human joints estimation. *Sensors*, 21:1684, 03 2021.

[35] Jingxiao Zheng, Xinwei Shi, Alexander Gorban, Junhua Mao, Yang Song, Charles R. Qi, Ting Liu, Visesh Chari, Andre Cornman, Yin Zhou, Congcong Li, and Dragomir Anguelov. Multi-modal 3d human pose estimation with 2d weak supervision in autonomous driving, 2021.

[36] Christian Zimmermann, Tim Welschehold, Christian Dornhege, Wolfram Burgard, and Thomas Brox. 3d human pose estimation in rgbd images for robotic task learning, 2018.