This page will be replaced with the actual title page.

## 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, $REPLACE_WE$ 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 smooth as $REPLACE_WE$ would like it to be. Especially, if humans are in positions that are unnatural pose estimation is not perfect and can lead to errors. This thesis is written in collaboration with SilverFit, a company that develops video games for rehabilitation purposes. SilverFit deals with unnatural poses due to injury or old age in a lot of cases. In this thesis, $REPLACE_WE$ collect different scenarios in which human pose estimation can fail. $REPLACE_WE$ then develop a method to record different exercises to compile a dataset with both clean and faulty data. Additionally, the data is augmented based on the confidence values of the joint of the pose. $REPLACE_WE$ then use the augmented dataset to train a model that can detect faulty joints in the pose with a confidence rating. Finally, $REPLACE_WE$ find approaches to improve the robustness of human pose estimation during streaming. Opening 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[19, 7, 23]. However, since human pose estimation is a difficult task, it 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 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.

One of the hampering factors for human pose estimation is the user itself. If the posture is not as the model expects then errors might occur.

This thesis is written in collaboration with SilverFit[1]. SilverFit 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. They are interested in a fault estimation system for human pose estimation in their games. This thesis aims to develop such a fault estimation system for human pose estimation in their games.

In this chapter, $REPLACE\_WE$ give an overview of different applications of human pose estimation and the challenges that are associated with it. $REPLACE\_WE$ focus on applications of human pose estimation at SilverFit and their desire for a fault estimation system for human pose estimation in their games. Then $REPLACE_{W}E$ discuss the problem that $REPLACE_{W}E$ are trying to solve and the research question that $REPLACE\_WE$ are trying to answer. Finally, $REPLACE_{W}E$ explore other approaches to the problem and how they differ from $REPLACE_{O}UR$ approach and how they influence the development of this project.

## 1.1 Research question

As mentioned earlier, a major problem with human pose estimation is that it is not possible to tell if the joints are faulty or not. This is a problem for SilverFit, as they want to be able to tell if the joints 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 joint is considered faulty if it is not in the incorrect position, i.e. the distance from the theoretical position is greater than a chosen threshold, or if it missing from the skeleton.

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

In this thesis, $REPLACE_WE$ first ask what problems occur during human pose estimation and what common error sources are. $REPLACE_WE$ aim to find which problems are the most common and which joints are most affected by the errors. This will help give an overview of the issues related to human pose estimation and help develop ways to detect these issues.

Once $REPLACE_WE$ know the issues that occur during human pose estimation, $REPLACE_WE$ aim to develop a method that can capture the camera stream in a way that allows $REPLACE_US$ to label the data according to the exercise and environment it was captured in. This will allow $REPLACE_US$ to create a dataset that can be used for future purposes.

Furthermore, $REPLACE_WE$ try to find if it is possible, given a joint, the RGB data, and the depth data, to determine if the joint is faulty or not using machine learning.

Finally, based on the result of the model $REPLACE_WE$ attempt to fix the faulty joints in the pose estimation to create a more robust human pose estimation model.

## 1.2 Process Pipeline

## 1.3 Fundamentals

In this section, $REPLACE_WE$ explain some of the fundamentals that are used during this thesis.

### 1.3.1 Human pose estimation

To interact with computers humans have come up with a plethora of methods. Ranging from early punch cards to modern touch screens, the methods have evolved to be more natural and intuitive. In recent years, the use of cameras to interact with computers has become more popular, since they require no physical contact and can make the use of computer systems seamless when developed properly.

In this section, $REPLACE_WE$ discuss some of the methods that have been used to extract the pose of a human from videos of different formats. $REPLACE_WE$ also discuss possible applications of human pose estimation. Finally, $REPLACE_WE$

In chapter 2 Human Pose Estimation Difficulties, $REPLACE_WE$ go into more detail about the method $REPLACE_WE$ used to extract the pose and what factors influence the result of the pose estimation.

We will go into more detail about some of the state-of-the-art human pose estimators for both RGB and RGBD data and even from point clouds in Section 1.4 Related Work.

#### Pose visualisation

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 skeleton. This is the most common way to visualise 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,

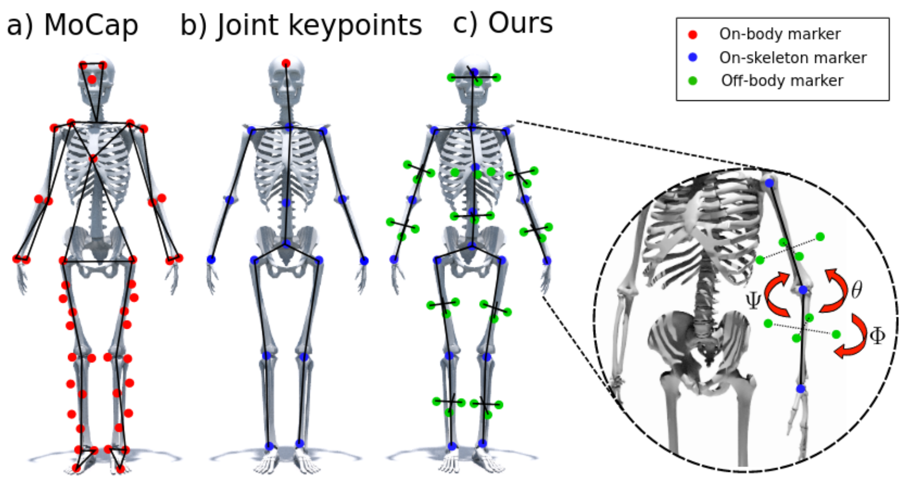


Figure 1-1: Example of a human pose captured with different methods. (a) A captured skeleton using MoCap which $REPLACE_WE$ do not focus on in this report. (b) A traditional human skeleton representation. (c) A pose representation that includes the orientation of the joints as well as the bones as presented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

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

### Pose estimation data sources

The pose of a human can be estimated from different types of data sources. The most common data source is RGB 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[20], and HumanEva-I dataset[27].

Additionally, some datasets are captured with depth cameras. Depth cameras are cameras that can record the depth of the 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].

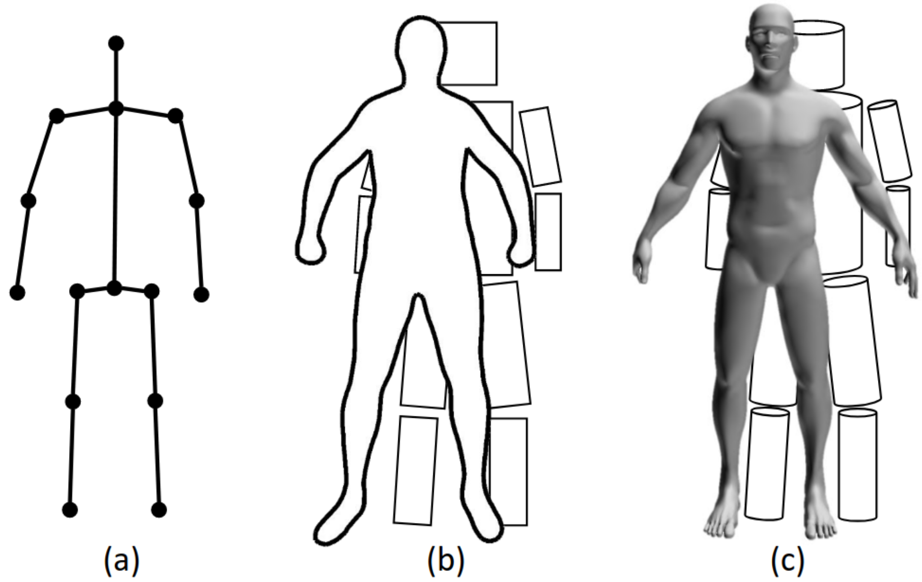


Figure 1-2: Different representations of the human pose. (a) Skeleton representation. (b) Contour representation. (c) 3D volume representation. [9]

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. 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.

Additionally, to the data that is provided, $REPLACE_WE$ also differentiate between monocular and multi-modal data. Monocular data is data that is captured with a single camera. Multi-modal data is data that is captured with multiple cameras. The most common multi-modal data is stereo data, which is data that is captured with two cameras. The cameras are usually placed next to each other and are angled toward the same scene. This allows the cameras to capture the same scene from different angles and therefore improving the accuracy of the pose estimation. There are not many datasets for this type of data, but the most common one was captured by Waymo[31].

However, RGB cameras are far more widely spread and generally cheaper than depth cameras. Hence, most methods use RGB cameras to estimate the pose of a human. However, since depth cameras can provide more detailed information about the scene, they can be used to improve the accuracy of the pose estimation and in this report, $REPLACE_WE$ will mainly focus on monocular RGBD data.

### Depth cameras

As mentioned earlier, human pose estimation generally works based on visual information. However, the use of depth cameras offers more detailed information about the scene, which can in turn improve the reliability of the pose estimation. Many different depth cameras function mainly on three different principles. Firstly there are stereo cameras. Stereo cameras try to calculate the depth of a scene similar to how human eyes work. Most of the time two lenses or cameras are placed or installed next to each other and are angled toward the same scene and then the depth of the scene is calculated by comparing the images captured by the two cameras. These cameras function on the spectrum of light which is visible to the human eye.

The second type of depth camera is the time-of-flight camera. These cameras use a laser to calculate the depth of the scene. The laser is fired at the objects in the scene, and the time it takes for the laser to bounce back is used to calculate the distance between the camera and the object based on the theoretical time it would take light to travel.

Finally, there are also structured light cameras. These cameras use a pattern of light that is known to the camera to calculate the depth structure of the scene. For both the time-of-flight and structured light cameras, the depth information is calculated based on the spectrum of light that is not visible to the human eye.

### Applications

Human pose estimation finds application in many different fields. Here $REPLACE_WE$ mention a few of the most common applications.

**Gaming and entertainment** This is 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 way. One of the games that kickstarted the use of depth cameras in games was the game Kinect by Microsoft. This game used a depth camera to track the movement of the player and used this information to control the game. This game was very successful and was used in many different games.

**Autonomous Driving** Autonomous driving has been in development ever since humans replaced horses with cars. 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 emulate exactly 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.

[2] Or animal

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.

**Healthcare** One of the companies that develop games using human pose estimation is Silverfit. SilverFit uses both a skeletal representation as well as a contour representation of the human pose. Human pose estimation enables the game to provide the user with feedback on their posture. This feedback can be used to improve the posture of the user and also allows the physiotherapists to design specific exercises which engage the muscles that are not used enough.

### 1.3.2 Machine Learning

### 1.3.3 Evaluation metrics and mathematical formulas

#### Precision and Recall

#### F1-Score

#### Cross-Entropy Loss

## 1.4 Related Work

**This still needs a lot of work**

A plethora of methods have been developed to estimate the pose of a human. In this section, $REPLACE_WE$ will discuss some of the methods that have been developed to estimate the pose of a human. Additionally, $REPLACE_WE$ discuss datasets that have been developed to test the performance of the methods. Finally, $REPLACE_WE$ discuss some of the methods that have been developed to estimate the fault.

### 1.4.1 Human Pose Estimation

Human Pose Estimation using Iterative Error feedback. [5]

While OpenPose developed Hand Pose[28] 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.

#### Reviews

A review of point cloud-based human pose estimation [34]

A review of 2D human pose estimation methods [10]

#### RGB Pose Estimation

But $REPLACE_W E$ wont go into much detail as $REPLACE_W E$ focus on RGBD data.

**This is a bit wrong, the dataset is multi modal but the definition of multi-modal is different in this method, it means RGB plus D and not different angles sometimes maybe not, read it through again:** The limited number of multi-modal datasets causes the existence of human pose estimators for cameras from different angles to be small. One example of multi-modal human pose estimation was introduced by Jingxiao Zheng et al.[35]. In their paper

#### RGBD Pose Estimation

**This is a bit out of context:** As mentioned by Jingxiao Zheng et al. in [35], the key points or joints of the skeleton do not lay on the surface of the person and therefore the determination of the exact position of the joints are not a direct projection on the depth image or the point cloud.

[22]

[36]

### 1.4.2 RGBD CNNs

Early HPE algorithm uses trees [26]

CNNs more useful for images and stuff. Cnns are not a new invention yada yada yada [13]. But like many things in the neural network Biz, they were limited by the hardware available at the time. They have since formed the basis of many new methods in computer vision, such as Human Pose estimation. Especially AlexNet [18] and VGG [29] proved the potential of CNNs in Computer Vision tasks.

### 1.4.3 Object Detection

[8] Proposes different methods of fusion for RGBD data.

#### Depth Completion

Realsense with Tensorflow [16] uses U-Net for depth completion [24].

#### Action Recognition

Cool CNN –¿ [11]

Another Review on human pose estimation but this time it is for action recognition [30]

In [25] Seddik et al. introduce different fusion methods for action recognition. They use RGB, Depth, and Skeleton data. After detecting the features for each modality, they fuse the features using different methods as can be seen in Figure **??**. Seddik et al. use different bags of visual words (BoVW).

Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments [17]

Latent Structured Models for Human Pose Estimation [6]

### 1.4.4 Anomaly Estimation

This is quite close to my topic I will find some papers that do this and write about them.

# Chapter 2

# Human Pose Estimation Difficulties

In this chapter, $REPLACE_WE$ discuss possible faults and difficulties that occur during human pose estimation. These difficulties are caused by different factors, such as the environment, the camera, the person, and the software. $REPLACE_WE$ discuss the most common difficulties and how they can be addressed.

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 first error source that $REPLACE_WE$ will discuss is the environment. $REPLACE_WE$ consider everything that is not the user or the camera as the environment. This includes the lighting of the room and the room itself. The environment can be an issue that is sometimes hard, if not impossible to fix.

In this section, $REPLACE_WE$ discuss the most common issues that occur in the environment and how they can be addressed.

### 2.1.1 Background

The background of the scene can cause difficulties in the human pose estimation process. In RGBD-based methods, the background can cause depth ambiguities, which might even prevent the human from being detectable. Also for depth cameras this can be an issue, however, depth ambiguities are less of an issue.

### 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 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, $REPLACE_WE$ refrain from experimenting with different lighting in the room. 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. Furthermore, $REPLACE_WE$ also discuss more general difficulties that can occur with any camera setup.

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 $REPLACE_WE$ mainly focus on pitch and roll. $REPLACE_WE$ assume that the user is in the centre of the camera, therefore, $REPLACE_WE$ do not regard yaw.

Most human pose estimators are trained without roll in mind. The cameras are usually setup so that any roll is minimised. To improve this human pose estimator can be trained with the data artificially rotated.

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.

**Need to create this** A sample of each exercise can be seen in Figure **??**.

#### 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 - Crossing the arms** Finally, the user will be standing and bowing forward.

**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 - 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.

**E-3.01 - Raising the knee leaning forward** Additionally to raising the knee, the user will now lean forward, to emulate bad posture.

**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. 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, $REPLACE_W E$ will discuss the different approaches that were considered and then $REPLACE_W E$ will discuss the approach that was chosen. $REPLACE_W E$ then discuss difficulties that were encountered during the data processing process.

As mentioned earlier, $REPLACE_W E$ propose 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, $REPLACE_W E$ have already discussed different methods that can be used for human pose estimation. In the scope of this thesis, $REPLACE_W E$ will only focus on a single human pose estimator. This has multiple reasons. Firstly, $REPLACE_W E$ 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*[26]. 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 first simplification is to consider each joint, i.e. the individual arms, legs, torso, and head. The second simplification is to consider the upper and the lower body. Finally, only the whole body is considered.

# Chapter 4

# Model development

While there could be multiple approaches to fault estimation, $REPLACE_WE$ have chosen to use a deep learning approach. 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. This is especially useful in $REPLACE_OUR$ case, as $REPLACE_WE$ have a large amount of data, but $REPLACE_WE$ do not know which features are important for the fault estimation.

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 4-2. 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 **??**. Since the application is in gaming a lightweight model which does not impact the performance much is preferred. Therefore, $REPLACE_WE$ compare the models by their number of floating-point operations (FLOPS) to their Accuracy on ImageNet-1K. Honorable mentions as well as the five most performing models are marked in the plot with their Top-5 and Top-1 Accuracy. Table **??** 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

$2.15 \times 10^7$ Parameters and a Top-1 Accuracy of 84.228%. The model architecture can be seen in Figure **??**. 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 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 4-3.

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 4-3 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 4-3 $REPLACE_WE$ 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 $REPLACE_WE$ have multiple objects, in $REPLACE_OUR$ case the different joints, $REPLACE_WE$ have to slightly change the loss function. If $REPLACE_WE$ were to use the Cross Entropy loss on the results as $REPLACE_WE$ got them now, the model would try to optimise toward a single class for all joints, which does not correctly reflect the desired result. Therefore, $REPLACE_WE$ apply the Cross Entropy loss on every joint and propagate them individually.

**Show formula of loss function**

## 4.4 Evaluation

Finally, $REPLACE_WE$ evaluate $REPLACE_OUR$ model by calculating different error metrics such as the root mean squared error as can be seen in Equation **??**, and the cross entropy loss. $REPLACE_WE$ also calculate the accuracy of the model, which is the percentage of correctly predicted faults. $REPLACE_WE$ also calculate the precision and recall of the model. The precision is the percentage of correctly predicted faults out of all predicted faults. The recall is the percentage of correctly predicted faults out of all faults in the data. $REPLACE_WE$ also calculate the F1 score, which is the harmonic mean of the precision and recall. The F1 score is a good indicator of the overall performance of the model.

**Explain Confusion Matrix**
**Explain Cohen Kappa Metric**
**Briefly explain Accuracy, Precision, Recall and F1**
**Explain Testing**

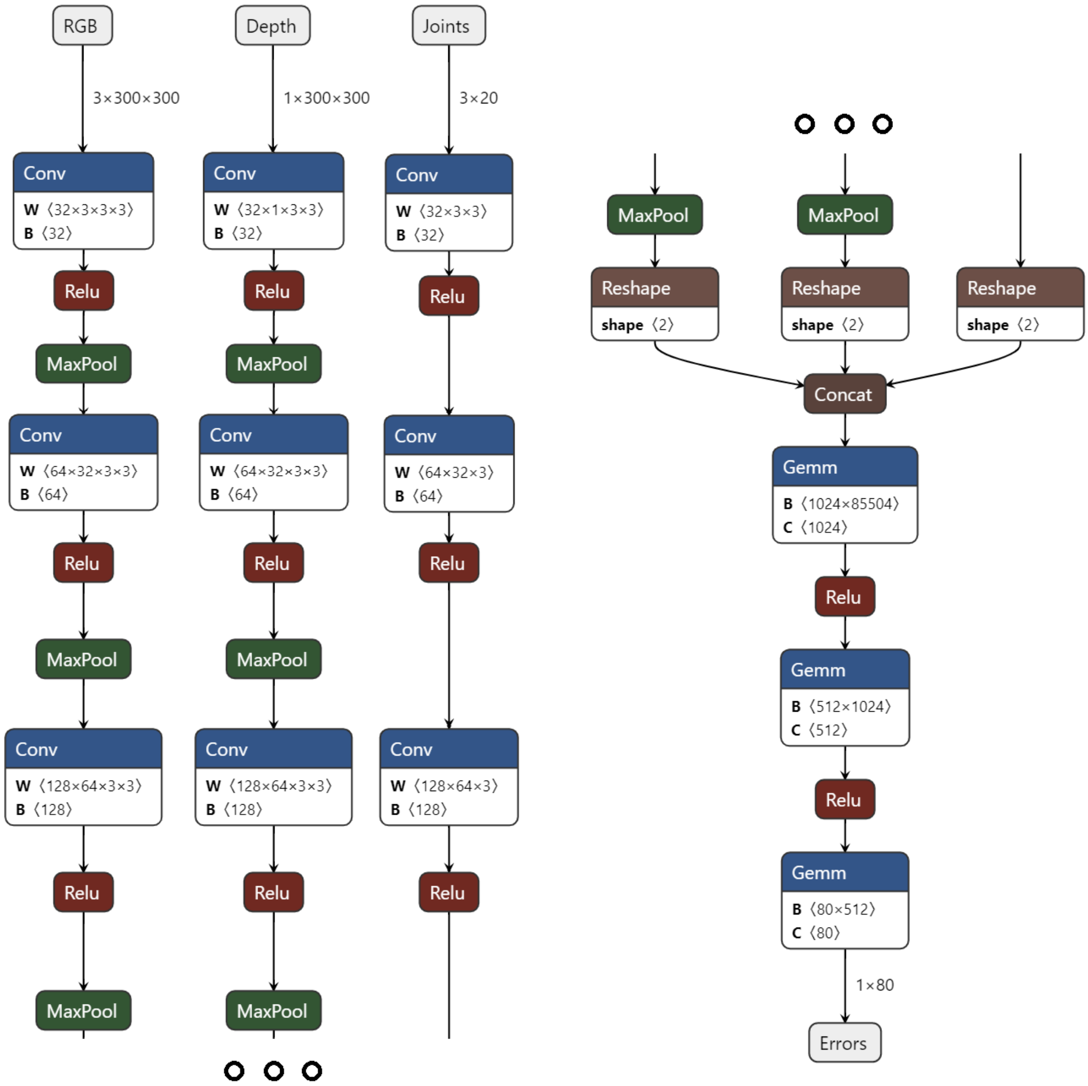


Figure 4-1: Original FESDModel architecture with three different inputs; RGB, Depth and Joint data. After three convolutions the three streams are concatenated to be passed into three fully connected layers. The output is a 1D 80 tensor of multi-object multi-class values.

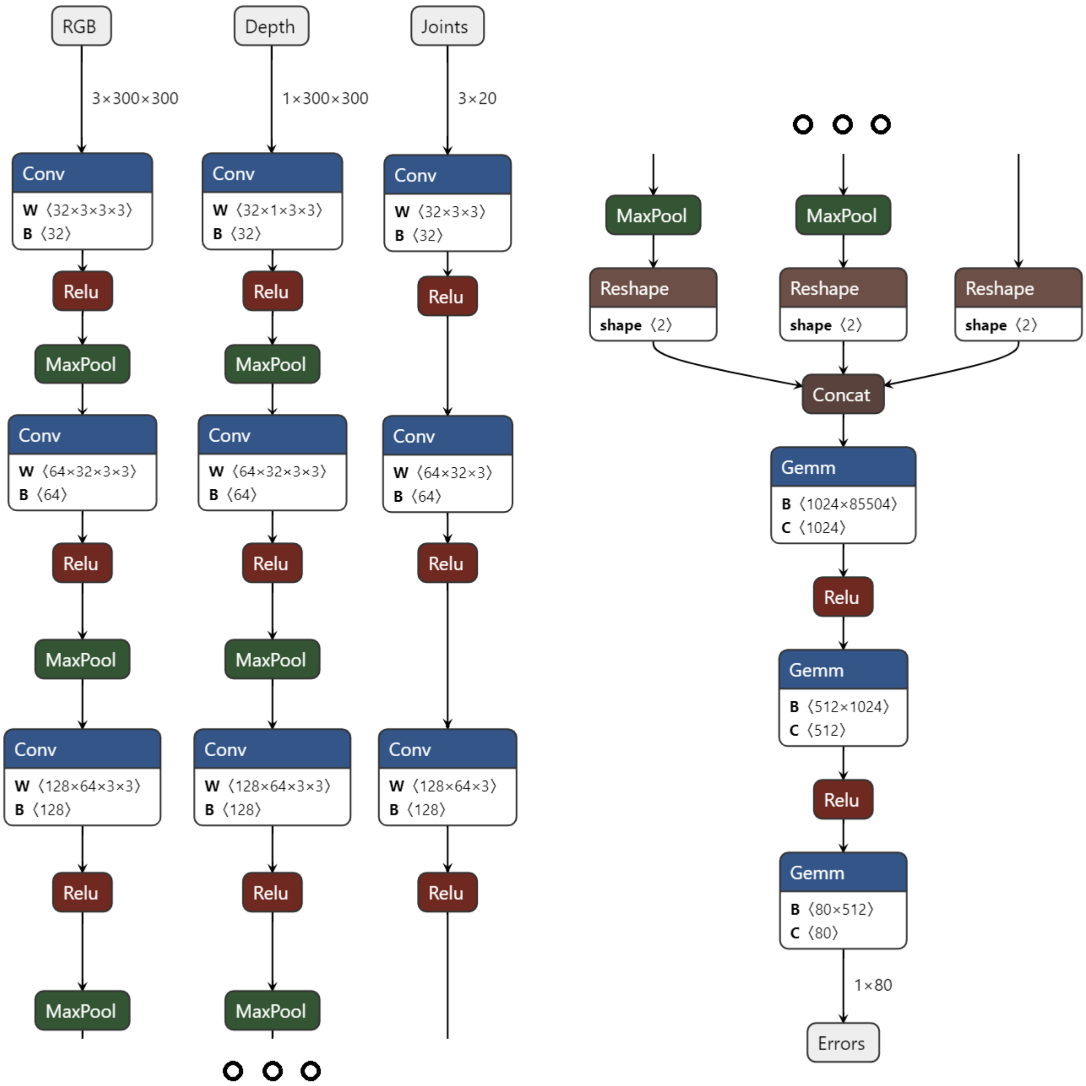


Figure 4-2: FESDModel architecture with transfer learning.

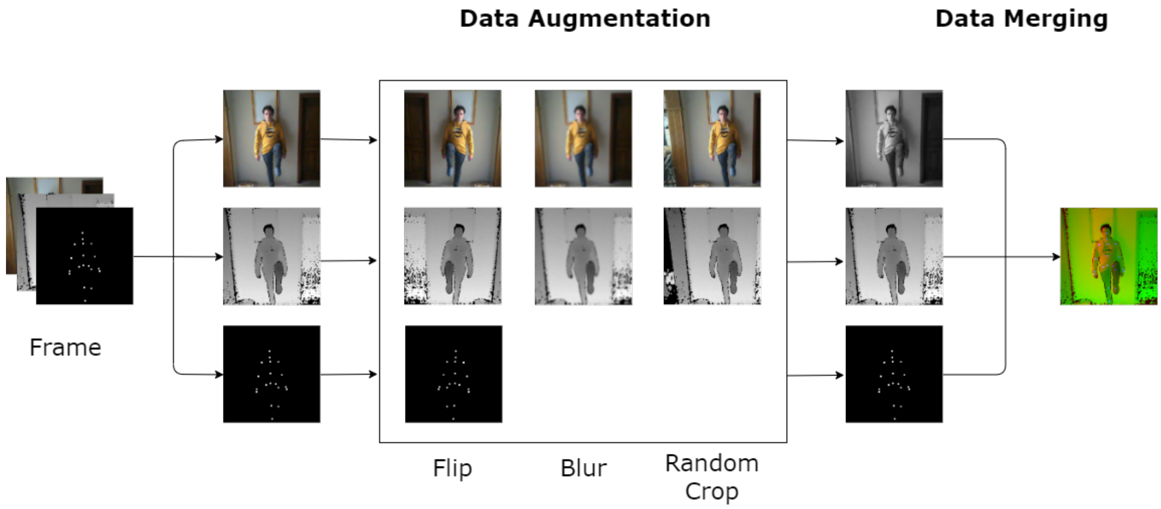


Figure 4-3: The data preparation pipeline. The data is randomly flipped, Blurred, and Cropped and then merged into a single RGB image.

# Chapter 5

# Experiment and Results

## 5.1 Experiment

- Cameras and how $REPLACE_WE$ used them
- The recording process
- Evaluation process evaluation, how many recordings were labeled
- How much data was used for training and how much for testing
- What hardware was used
- How many epochs, what training rate and all those parameters

## 5.2 Results

- What results are $REPLACE_WE$ expecting and why
- Graphs with different metrics
  - Confusion matrix for each difficulty
  - Each metric mentioned in model evaluation
- Evaluation and explaination of the graphs
- Have $REPLACE_WE$ achieved what $REPLACE_WE$ were looking for?

# 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`

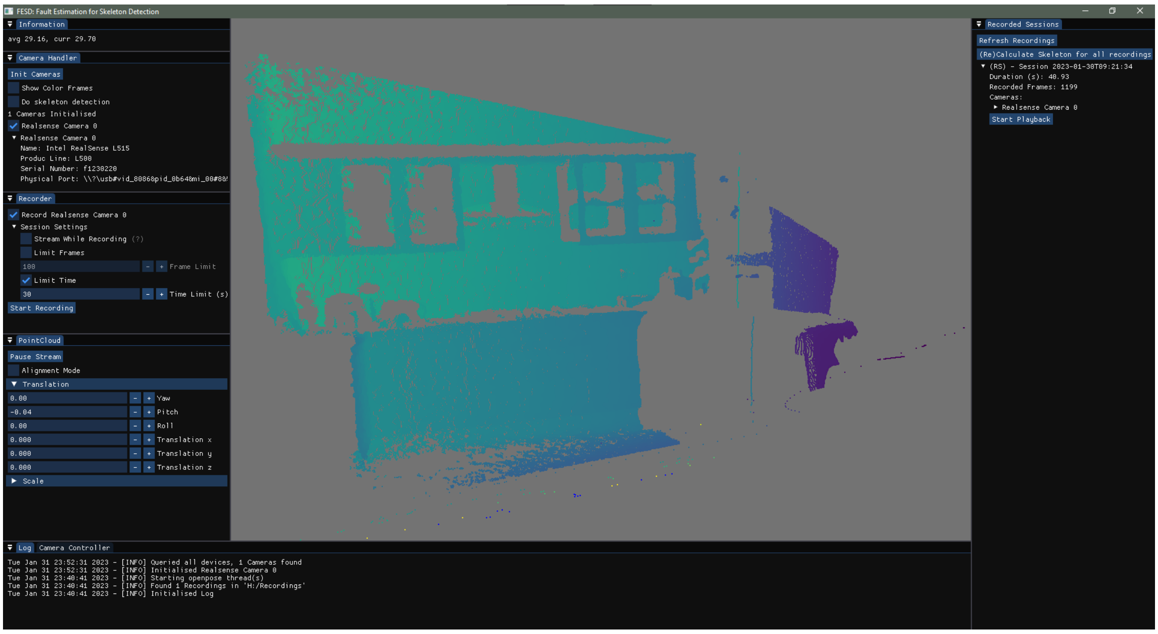


Figure 6-1: A screenshot of the FESDData GUI streaming a point cloud. The GUI is used to record, playback, visualise and label the data streams from RGBD cameras. The FESDData Gui is written in C++ using the ImGui framework. The point cloud is visualised using OpenGL and a glsl shaders.

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

There are a multiple areas in which FESD can be expanded. Firstly, FESDData is already quite stable and user friendly software, however, as any software it can be improved. Firstly, the file storage is efficient enough for data recording but it is prone to error and can easily be interrupted by a corrupt file. In a future, more stable version of FESDData storing each recording in a separate RosBag file would improve the stability of the file system. Additionally, $REPLACE_WE$ have not fixed some edge case issues which were not essential to the thesis itself.

The developed model could be improved by increasing the training data and recording in different settings. To further validate the model $REPLACE_WE$ could imagine applying it to a different dataset that is used by an action detector and evaluating its performance with introduced errors. Additionally, $REPLACE_WE$ could derive with a model which not only determines if there is an error, but also tries to fix the error by suggesting a different location.

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

[20] 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.

[21] 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.

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

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

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

[30] 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.

[31] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset, 2019.

[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.