This page will be replaced with the actual title page.

Abstract

The interaction between humans and computers has been a topic of interest for many years. 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 smooth as 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, we collect different scenarios in which human pose estimation can fail. 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. We then use the augmented dataset to train a model that can detect faulty joints in the pose with a confidence rating. Finally, we find approaches to improve the robustness of human pose estimation during streaming.

Contents

1 Introduction **1**
- 1.1 Human pose estimation . . . 1
  - 1.1.1 Pose visualisation . . . 2
  - 1.1.2 Pose estimation data sources . . . 3
  - 1.1.3 Depth cameras . . . 4
  - 1.1.4 Applications . . . 4
- 1.2 Research question . . . 5
- 1.3 Process Pipeline . . . 6
- 1.4 Related Work . . . 6
  - 1.4.1 Human Pose Estimation . . . 7
  - 1.4.2 RGBD CNNs . . . 7
  - 1.4.3 Object Detection . . . 8
  - 1.4.4 Fault Estimation . . . 8

2 Human Pose Estimation Difficulties **9**
- 2.1 Environment . . . 9
  - 2.1.1 Lighting . . . 9
  - 2.1.2 Objects . . . 10
  - 2.1.3 Chair . . . 10
- 2.2 Camera . . . 10
  - 2.2.1 Distance . . . 10
  - 2.2.2 Angle . . . 10
  - 2.2.3 Resolution . . . 10
  - 2.2.4 Depth Range . . . 10
- 2.3 Person . . . 10
  - 2.3.1 Clothes . . . 10
  - 2.3.2 Training Equipment . . . 11
  - 2.3.3 Exercises . . . 11

3 Data Processing **15**
- 3.1 Stream pre-processing . . . 15
  - 3.1.1 Multiple Cameras . . . 15
  - 3.1.2 Recording session set-up . . . 15
- 3.2 Data acquisition . . . 17
  - 3.2.1 Data format . . . 17
  - 3.2.2 Recording process . . . 20
- 3.3 Data population . . . 20

3.3.1 Human Pose Estimators . . . 21
3.3.2 Human Pose Estimation . . . 21
3.4 Data Evaluation . . . 21
3.5 Data Augmentation . . . 22

4 Model development **23**
4.1 Model training . . . 23
4.1.1 Data preparation . . . 23
4.1.2 Model Architecture . . . 23
4.2 Model evaluation . . . 23

5 Experiment **25**
5.1 Camera Setup . . . 25

6 Results **27**

7 Conclusion **29**
7.1 Contribution . . . 29
7.1.1 Developed Software . . . 29
7.1.2 Developed Model . . . 29
7.1.3 Possible applications . . . 29
7.2 Future work . . . 29

List of Figures

1-1 Example of a human pose captured with different methods. (a) A captured skeleton using MoCap which 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[**?**] 2
1-2 Different representations of the human pose. (a) Skeleton representation. (b) Contour representation. (c) 3D volume representation. [**?**] 3
1-3 Process Pipeline with all steps 6

3-1 Pinhole camera model 18
3-2 Recording GUI marked in Red 21

5-1 Accelerometer data from the Realsense camera used to set up the camera. . 26

7-1 FESD GUI 30

List of Tables

Listings

3.1 Example of session metadata 18
3.2 Pseudo code for data evaluation 22

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. 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, we give an overview of different applications of human pose estimation and the challenges that are associated with it. 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 we discuss the problem that we are trying to solve and the research question that we are trying to answer. Finally, we explore other approaches to the problem and how they differ from our approach and how they influence the development of this project.

1.1 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, we discuss some of the methods that have been used to extract the pose of a human from videos of different formats. We also discuss possible applications of human pose estimation. Finally, we

In chapter 2 Human Pose Estimation Difficulties, we go into more detail about the method 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.

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

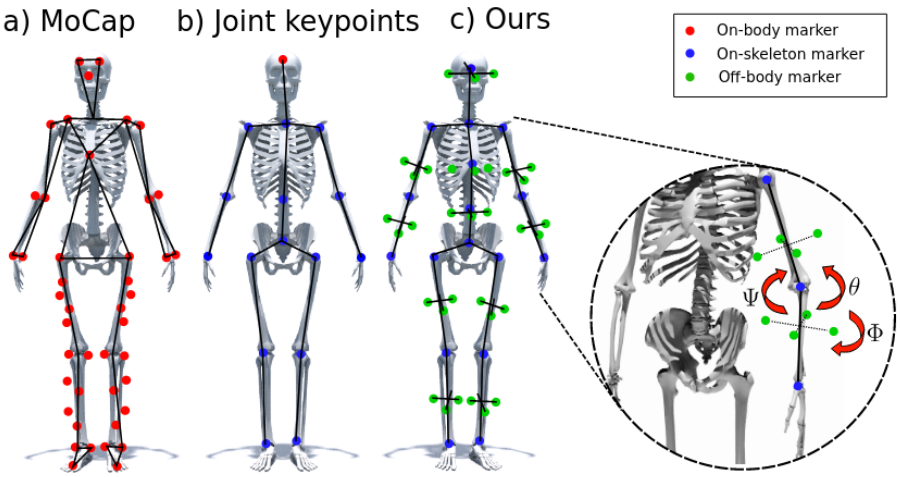

Figure 1-1: Example of a human pose captured with different methods. (a) A captured skeleton using MoCap which we do not focus on in this report. (b) A traditional human skeleton representation. (c) A pose representation that includes the orientation of the joints as well as the bones as presented by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

1.1.1 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[**?**]. 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[**?**]. 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.

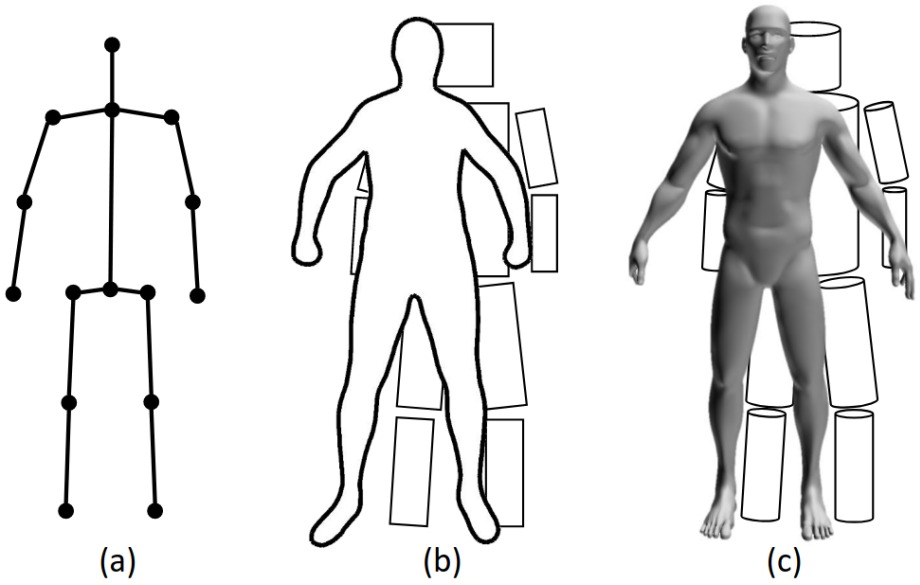

Figure 1-2: Different representations of the human pose. (a) Skeleton representation. (b) Contour representation. (c) 3D volume representation. [?]

1.1.2 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[?], the COCO dataset[?], and HumanEva-I dataset[?].

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[?], and the Human3.6M dataset[?].

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[?], and the EVAL dataset[?].

Additionally, to the data that is provided, 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[?].

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, we will mainly focus on monocular RGBD data.

1.1.3 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.

1.1.4 Applications

Human pose estimation finds application in many different fields. Here 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.

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.

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.

1.2 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.

In this thesis, we first ask what problems occur during human pose estimation and what common error sources are. 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 we know the issues that occur during human pose estimation, we aim to develop a method that can capture the camera stream in a way that allows us to label the data according to the exercise and environment it was captured in. This will allow us to create a dataset that can be used for future purposes.

Furthermore, 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 we attempt to fix the faulty joints in the pose estimation to create a more robust human pose estimation model.

[2]Or animal

1.3 Process Pipeline

The whole process of fault estimation can be seen as a pipeline. We start at the most basic starting block, the camera streams, and end at the most complex block, the fault estimation. The pipeline is shown in Figure 1-3. The pipeline consists of eight steps, which are described in more detail in the following sections. The steps are (**0**) Preliminary Analysis, (**I**) Stream Pre-Processing, (**II**) Data Acquisition, (**III**) Data Population, (**IV**) Data Post-Processing/Evaluation, (**V**) Data Augmentation, (**VI**) Model Training, and (**VII**) Model Evaluation. The results of each step are used as input for the next step.

We further divided the process into a preliminary, data processing, and model development phase. The preliminary phase focuses on issue analysis and exercise development, the data processing phase is the first five steps of the pipeline. The model development phase is the last two steps of the pipeline.

In the next chapters, we give a basic overview of the whole process. We go into more detail in the following sections.

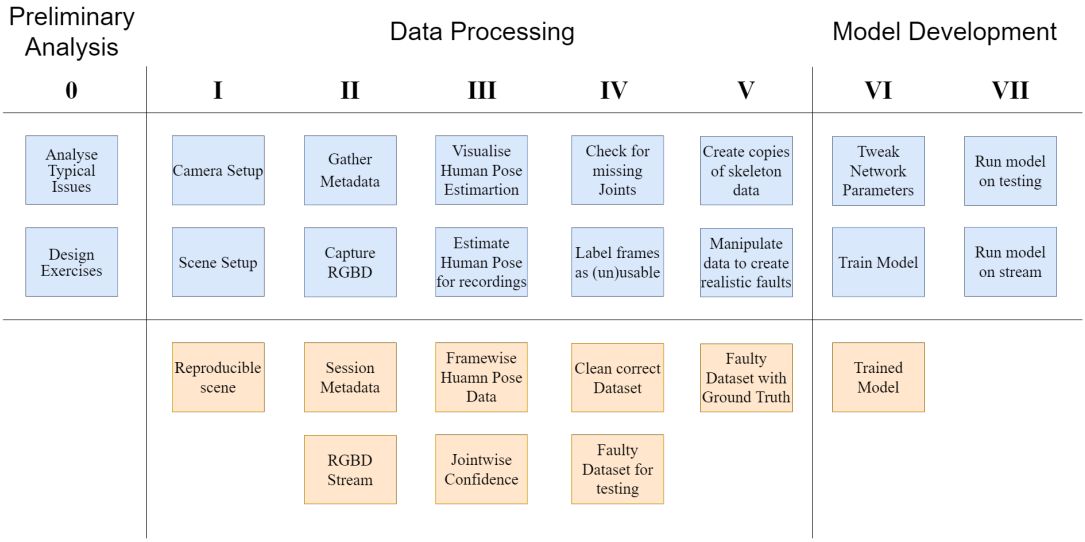

Figure 1-3: The whole Process pipeline with all the steps, which are marked in blue, and the results of each step, which are marked orange. The results of the steps are used as input for the next step. The steps are described in more detail in the following sections. The steps are: (**0**) Preliminary Analysis, (**I**) Stream Pre-Processing, (**II**) Data Acquisition, (**III**) Data Population, (**IV**) Data Post-Processing/Evaluation, (**V**) Data Augmentation, (**VI**) Model Training, and (**VII**) Model Evaluation.

1.4 Related Work

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

of the methods. Finally, 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. [?]

While OpenPose developed Hand Pose[?] and also Multi-Person Human Pose Estimation [?], our main focus lies on their most recent pose estimator [?] and their CNN network [?]. 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 [?]

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

RGB Pose Estimation

[?]

But we wont go into much detail as we 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.[?]. In their paper

RGBD Pose Estimation

This is a bit out of context: As mentioned by Jingxiao Zheng et al. in [?], 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.

[?]

[?]

Nuitrack does not offer any white paper or documentation on their method. However, they have written that they use a CNN to estimate the pose of a human. That CNN uses both RGB and depth information to estimate the pose of a human.

1.4.2 RGBD CNNs

Early HPE algorithm uses trees [?]

CNNs more useful for images and stuff. Cnns are not a new invention yada yada yada [?]. 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 [?] and VGG [?] proved the potential of CNNs in Computer Vision tasks.

1.4.3 Object Detection

[**?**] Proposes different methods of fusion for RGBD data.

Depth Completion

Realsense with Tensorflow [**?**] uses U-Net for depth completion [**?**].

Action Recognition

Cool CNN –¿ [**?**]

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

Great input fusion graphic [**?**]

1.4.4 Fault Estimation

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

Latent Structured Models for Human Pose Estimation [**?**]

Chapter 2

Human Pose Estimation Difficulties

In this chapter, 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. 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 we will discuss is the environment. 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, we discuss the most common issues that occur in the environment and how they can be addressed.

2.1.1 Lighting

Most RGBD cameras use infrared light to determine the depth of the scene, some use a pattern of infrared light which 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.

Solution

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, we refrain from experimenting with different lighting in the room. Therefore, we do all of the recordings in a room without any natural light or at night.

2.1.2 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.

2.1.3 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 SilverFits 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.

2.2 Camera

In this section, 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, we also discuss more general difficulties that can occur with any camera setup.

The two main difficulties that can occur with the camera are the camera position and the camera angle.

2.2.1 Distance

2.2.2 Angle

2.2.3 Resolution

2.2.4 Depth Range

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, we define some exercises that are easy to detect as well as some exercises that are difficult to detect. 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 sedentry 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. [**?**]

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

In this chapter, we discuss the data processing steps that are required to prepare the data for the model development process. Firstly, we address different session parameters and how they might influence the data. Secondly, we explain the data acquisition process and how the data is stored in files such that it can be used in the future. Thirdly, we discuss the data population process in which the human pose data is extracted from the raw data. To ensure the quality of the dataset we then evaluate the data by filtering invalid skeletons and marking data points as valid or invalid. Finally, we discuss the data augmentation process in which the data is augmented to increase the size of the dataset.

3.1 Stream pre-processing

To get the best possible results we need to make sure that the cameras are set up in exactly the intended way.

3.1.1 Multiple Cameras

We use multiple cameras to increase the accuracy of the results. We use two cameras to record the same scene from two different angles. This way we can compare the results from the two cameras and make sure that the results are consistent. We also use multiple cameras to record the same scene from different heights and angles.

UNSURE *Should I write about it if Im not going to do it? Its quite interesting how the synchronisation might work and how the pointclouds can be synchronised. I already did a lot of research on it but if Im not going to implement it then this might not be the best point to do it.*

3.1.2 Recording session set-up

We consider different environmental setups to increase the significance of the results. The following session parameters are considered:

Lighting

RGBD cameras function with infrared light therefore is the lighting of a scene essential. We found that direct sunlight interferes with some RGBD cameras more than others based on the infrared range that is used. Since the exact sunlighting is not controllable we choose

to make it as optimal as possible to improve reproducibility. Therefore, we choose a room with no sunlight but we do include artificial light to reduce any damage that might occur to visibility issues.

Relative Camera Position

At SilverFit, cameras are attached above a screen at a height of $180cm$ facing downward at around 20 deg. To form a more general model, we will experiment with different setups and angles. We experiment with six different setups in total. Three setups from different angles ($20\,\text{deg}, 0\,\text{deg}, 340\,\text{deg}$) at two different heights ($180cm$, $120cm$). The different setups can be seen in figure TODO.

TODO *Add figure with different setups.*

UNSURE *We Develop a functionality that lets us determine the exact height and orientation of the camera. We do this by detecting the floor and thereby calculating the height of the camera and the angle at which it is pointing downward. We can also detect if the camera is not completely straight and therefore might influence the results.*

Sitting or standing

From experience, we know that detecting the joints correctly is influenced by the position of the participant. This is especially true for the difference between a sitting and a standing patient. Human pose detection is in general more reliable if the patient is standing, due to reduced occlusion. We record each scenario sitting and standing.

Clothing and ankle and wrist attachments

Clothing can have a similar effect on the efficacy of HPE as lighting. If the participant is wearing black pants infrared light will be absorbed rather than reflected leading to 'blind spots' in the legs. Since the legs are already more unreliable than the rest of the body, these blind spots can negatively affect HPE.

Since SilverFit develops games for rehabilitation, the supervising physiotherapist might choose to attach weights to the ankles and/or wrists to increase the effectiveness of the exercise. We therefore also include attached and held weights to simulate difficult situations.

Background

The background of the scene can have a significant effect on the results. We, therefore, record the same scenario with and without a visible background, i.e. a wall is behind the participant or there is no wall within the maximum sensor range ($6m$).

Crampedness of the Environment

The Crampedness of the scene increases the number of false positives of HPE. We, therefore, record the same scenario with and without clutter. We consider clutter to be any object that is not a part of the participant's body. However, clutter is quite objective and therefore we will not be able to define it in a universally applicable way.

Distance to the camera

Games developed by SilverFit have a calibration step where the participant is asked to stand at a certain distance from the camera. We, therefore, record the scenario at that specific distance. This ensures that noise introduced by the depth sensor has little effect on the results. The participant is positioned 2 meters away from the camera. *UNSURE, ask someone at SilverFit.*

3.2 Data acquisition

The second phase of the pipeline is data acquisition. After we have set up the camera according to the session parameters we can start recording. We set a timer for $30s$ and record the RGB and Depth streams from the camera. We also record the camera intrinsics, which we will use to recreate the point cloud at a later stage. We record the data in a folder structure that is defined by the session parameters.

3.2.1 Data format

An important part of data acquisition is the description of the way the data is stored in the file system. This is essential for any future use of the data and therefore we need to make sure that the data is stored in a way that is easy to understand and easy to use. We store in general two to five files per session depending on the camera configuration.

Session Metadata

Every session contains a "`SESSION_NAME.json`" file that contains the session parameters and camera metadata. The session name is automatically generated based on the starting time of the session, this way we can make sure that the session name is unique every time and we can also have an idea of which recording is the most recent without looking at the contents of the file.

Camera metadata The camera metadata contains the camera intrinsics, which we will use to recreate the point cloud at a later stage. The camera intrinsics are the field of view of the depth camera in the horizontal and vertical direction, and the principal point in the horizontal and vertical direction. The field of view is the angle between the optical axis and the image plane. The principal point is the point in the image where the optical axis of the camera intersects the image plane. The principal point and the field of view are explained in Figure 3-1.

Camera orientation Additionally, to the camera intrinsics we store the relative rotation and translation between the cameras if multiple cameras are used. The rotation and translation are stored as Euler angles[1] and a vector respectively [**?**]. The rotation is the rotation of the camera relative to the second camera in the system. The translation is the translation of the camera concerning the second camera in the system.

[1] Technically we are storing the rotation with the Tait–Bryan notation, i.e. x-y-z or yaw-pitch-roll, rather than the classic Euler notation. However, the name Euler angle is more commonly used and understood.

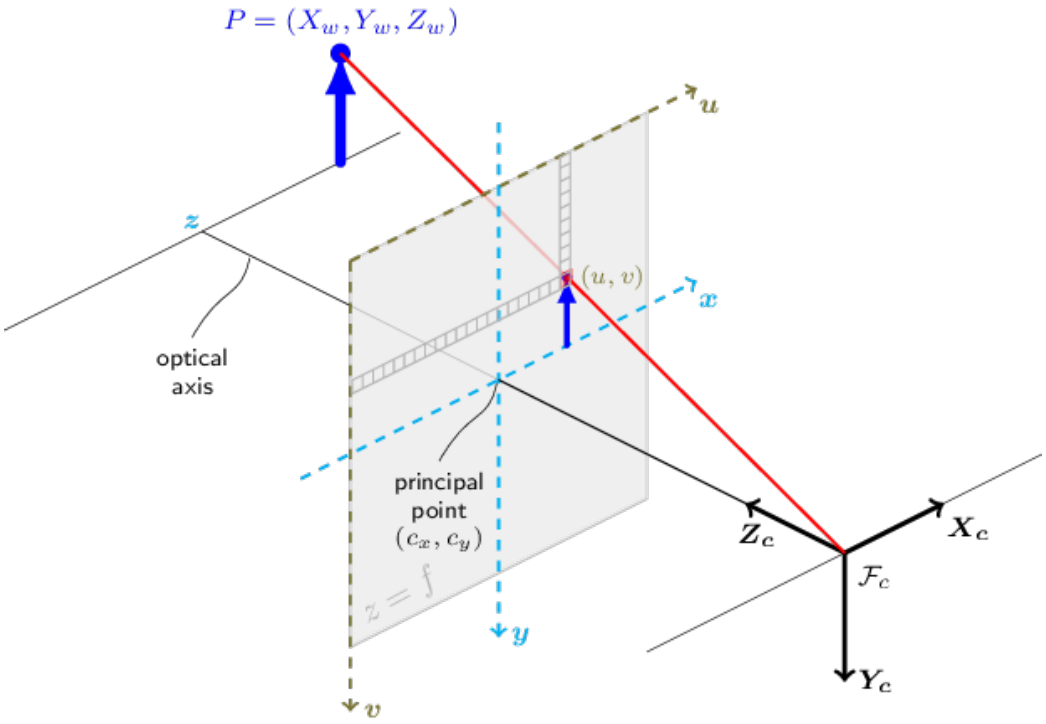

Figure 3-1: The pinhole camera model that shows the principal point. The principal point is the point where the optical axis intersects the image plane. The field of view is the angle between the optical axis and the image plane.

Session parameters The session parameters are the same as the ones defined in the section "Stream pre-processing". The user enters the session parameters before starting the recording. Most of the parameters are boolean values that indicate whether the user is sitting, wearing dark clothing, etc. The height and angle parameters are the height of the camera to the floor and the angle of the camera relative to the orientation of the user as explained in the previous section.

An example of the Session metadata can be seen in Listing 3.1.

Listing 3.1: Example of the Session metadata with a single Realsense Camera which was recorded for 40 seconds at around 30 frames per second resulting in 1200 frames. Some values have been changed to increase readability.

```
{
  "Cameras" :
  [
    {
      "Cx" : 314.26,
      "Cy" : 239.46,
      "FileName" : "Session_2023-01-30T09.21.34_Realsense_Camera_0.bag",
      "Fx" : 459.77,
      "Fy" : 459.83,
      "MeterPerUnit" : 0.00025,
      "Name" : "Realsense Camera 0",
      "Type" : "Realsense"
    }
```

```
  ],
  "DurationInSec": 40.0,
  "Name": "Session 2023-01-30T09:21:34",
  "RecordedFrames": 1200,
  "Rotation": {
      "Roll": 0.0,
      "Pitch": 0.0,
      "Yaw": 0.0
  },
  "Translation": {
      "X": 0.0,
      "Y": 0.0,
      "Z": 0.0
  },
  "Session Parameters": {
    "Sitting": true,
    "Background close": true,
    "Cramped": false,
    "Dark Clothing": true,
    "Holding Weight": false,
    "Ankle Weight": false,
    "Height": 1.8,
    "Angle": 20.0
  }
}
```


Realsense Cameras

We record Realsense Cameras using the librealsense SDK provided by Intel. Using the SDK we have access to the Hight-Level Pipeline API which allows us to stream the camera feed and access the camera intrinsics. This High-Level Pipeline API allows us to record the RGB and Depth streams from the Realsense camera. The SDK automatically synchronises the Depth and RGB stream as well as the motion sensors, which we do not use since our camera is static. The librealsense SDK is available on GitHub[2].

The Recordings are stored in a ROS bag file. A ROS bag file is a file format for storing ROS messages. The ROS bag file format is a container format that stores multiple messages in a single file. The ROS bag file format is described in detail in the ROS wiki[3]. The ROS bag file format is a container format that stores multiple messages in a single file. In our case, the important messages are the camera intrinsics, which allow us to create a virtual Realsense Camera from the recording, the RGB stream, and the Depth stream. However, other messages are also stored and can be accessed using the ROS Bag API[4].

Orbbecc Astra Cameras

To read the depth stream of the Orbbecc Astra camera we use the OpenNI2 API[5]. The OpenNI2 API is a cross-platform API that allows us to access the depth stream of the Orbbecc Astra camera. The OpenNI API is no longer being developed by PrimeSense and

[2] https://github.com/IntelRealSense/librealsense
[3] http://wiki.ros.org/Bags
[4] http://wiki.ros.org/rosbag/Code%20API
[5] https://structure.io/openni

has been renamed to OpenNI2 to avoid confusion with the OpenNI API. The OpenNI2 API is available on GitHub[6].

Using the OpenNI2 API we can also record the depth stream to a file. The depth stream is stored as a `.ONI` file. The `.ONI` file format is a proprietary format that is not documented. However, the OpenNI2 API provides a `.ONI` file reader that allows us to access the depth stream.

Sadly, the OpenNI2 API does not provide a way to access the RGB stream. Therefore, we use the OpenNI2 API to access the depth stream and OpenCV to access the RGB stream. The RGB stream is stored as a `.AVI` file. The `.AVI` file format is a container format that stores multiple video streams in a single file. The `.AVI` file format is described in detail in the Microsoft documentation[7].

***ISSUE**: currently the playback of the `.AVI` file is only possible at a specific framerate, which is set at the beginning of the recording session. This poses a substantial issue regarding synchronisation. Should I write about this? Should I only use depth data?*

3.2.2 Recording process

Once the scene is set and the point clouds have been aligned the user can start the recording. Firstly, there are pre-recording settings such as a frame and/or time limit in seconds, which allows accurate time and/or frame constraints to create equal recordings. Secondly, the user can select whether or not to display the point cloud while recording. This allows the user to see the point cloud in real-time and adjust the recording accordingly, however, this leads to a substantially reduced framerate[8]. These can be set in the GUI as shown in Figure 3-2. Finally, the user can configure the session parameters described earlier. After the recording has beend started by the user a preconfigured countdown will be started, allowing the user to get into position in time. Once the countdown has finished the recording will start. The recording will stop once the time or frame limit has been reached. The recording will also stop if the user presses the stop button.

3.3 Data population

***JUST AS REFERENC** To achieve the highest framerate, we calculate the skeleton based on the recorded data and add it to the dataset in a separate step. The RGB stream is used to create the dataset for the skeleton detection and the depth stream is used to create the point clouds for the calculation of global skeleton points. We store both the local 2D coordinates in accordance with the image used for the skeleton detection, as well as the global 3D coordinates based on the aligned point clouds. Additionally, OpenPose provides us with a confidence score for each joint.*

***DECISION** I decided to switch to NuiTrack, it is closer to silverfit and I think a better choice. Openpose poses more problems than it solves*

***TODO** Explain what is meant by Data Population (skeleton detection).*

[6] `https://github.com/structureio/OpenNI2`

[7] `https://docs.microsoft.com/en-us/previous-versions/ms779636(v=vs.85)`

[8] From 30 FPS, without point cloud to 15 FPS with pointcloud

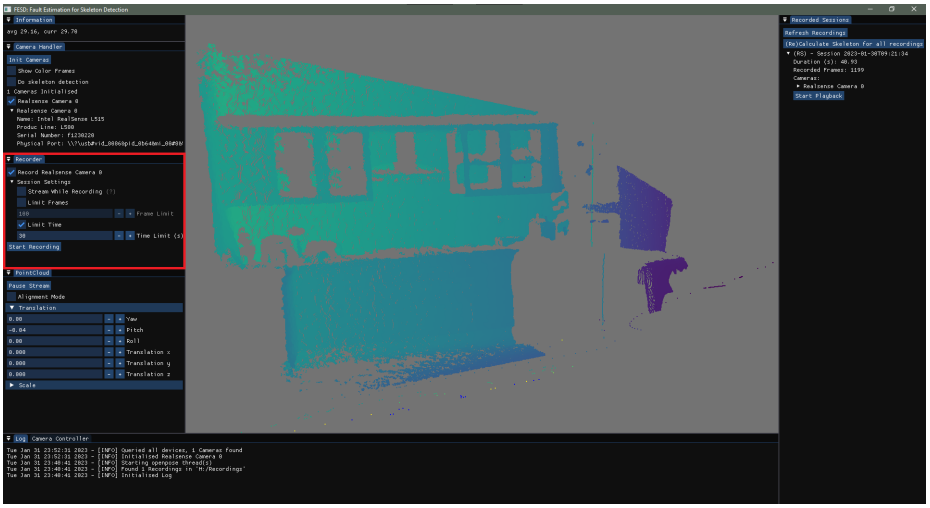

Figure 3-2: Recording GUI marked in Red. The user can set the recording parameters here such as the time and frame limits and whether or not the .

3.3.1 Human Pose Estimators

OpenPose

TODO *Give rough overview of Openpose and how the projection might have worked.*

NuiTrack

To utilise skeleton data, as well as the human silhouette in their games, SilverFit utilises the NuiTrack SDK. ***TODO*** *Explain what NuiTrack is and does and how it works.*

3.3.2 Human Pose Estimation

Human pose estimation is not trivial in terms of resource usage. Calculating the human pose while recording would mean a significant reduction of the recorded frames. Therefore, we calculate the skeleton seperatly for each frame of the recording.

The skeletons is stored ... ***TODO*** *Explain how the skeleton is stored.*

3.4 Data Evaluation

Next, we evaluate the recorded data and especially the detected skeleton. We focus specifically on the joints with a low confidence value.

We discard frames with lacking pose data from the training dataset, they are not usable to train a model. We might still use it for the testing phase. If we notice that the dataset is getting too small, we might re-record some sessions.

QUESTION *Should I discard frames with limited joints?*

Additionally, while recording multiple people might be wrongly detected. However, in our experiments we only consider single person recordings. Therefore, we can discard other people from the data.

The pseudo code for the data evaluation process can be seen in Listing 3.2. Most of the checks mentioned, such as `selectInvalidPeople` in Line 9 or `checkJointValidity` in

Line 14 happen manually. However, the data processing is done automatically by the code to redue human error.

Listing 3.2: Pseudo code for data evaluation

```
def data_evaluation(recording):
  invalid_frames = []
  for frame in recording:
    if not frame:
      invalid_frames.append(frame)
      continue
    else:
      if len(frame.people) > 1:
        invalid_people = frame.selectInvalidPeople()
        frame.remove(invalid_people)

      for joint in frame.people[0].skeleton:
        if joint.confidence < CONFIDENCE_THRESHOLD:
          checkJointValidity(joint)

  if len(invalid_frames) > INVALID_LIMIT:
    return False

  recording.replace(invalid_frames, Null)

  return True
```

3.5 Data Augmentation

Once the data is cleaned and we filter recordings with too many missing joints, we can augment the data to simulate a larger amount of data with the ability to create faulty scenarios controlled. One major fault is the seemingly random detachment of the joint to a side. This especially affects the legs and arms, therefore we will have a bias toward limbs with this augmentation.

Furthermore, we randomly move the joints with low confidence. Another fault is the disappearance of joints. We use the same bias as with the random detachment, i.e. we take the limb bias, as well as the confidence into consideration.

This phase allows us to create a large amount of data with a controlled amount of faults. This is important since we want to be able to train a model that can detect faults in the data. The augmented data is stored in a separate file so that we can use it to train the model and compare it to the original manually checked ground truth.

TODO *Create some screenshots of the augmented data from the different methods.*

Chapter 4

Model development

While there could be multiple approaches to fault estimation, 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 our case, as we have a large amount of data, but 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 training

Using this enlarged dataset, we can train a Neural Network to recognise faults in the data. We use the depth data as input, the skeleton data as input, and a combination of both as input. We also experiment with different network layouts, such as a fully connected network, a convolutional network, and a combination of both. We use the augmented data to train the model and the manually checked ground truth to validate the model. We use the validation data to determine the best model and the best network layout. We use the best model to predict the faults in the data.

4.1.1 Data preparation

I guess this is the part where I should explain how the data is prepared for training. I think I should explain the data augmentation and the data splitting. I should also explain how the data is prepared for the different network layouts.

4.1.2 Model Architecture

4.2 Model evaluation

Finally, we evaluate our model by calculating different error metrics such as the mean absolute error, the mean squared error, and the root mean squared error. We also calculate

the accuracy of the model, which is the percentage of correctly predicted faults. 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. 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.

Chapter 5

Experiment

- Cameras and how we used them
- Setup check (is the setup correct?)
- The recording process
- What Exercise was chosen and why
- Data size (not that important but still good to give perspective)
- Evaluation process evaluation, how many recordings had to be discarded, how many frames were invalid, and so on
- How many seconds and frames were recorded in total
- How much data had to be discarded due to missing ground truth
- How much data was used for training and how much for testing

5.1 Camera Setup

The Realsense camera has an accelerometer. We know the target angle and current orientation. We can use this to fix the camera accordingly.

$$\frac{y}{x+y+z} * 90^\circ = Angleindegree$$

E.g.:

$$Angle = 70 \Rightarrow \frac{y}{x+y+z} * 90^\circ = 70^\circ where x = 0 \Rightarrow \frac{y}{y+z} = \frac{70}{90}$$

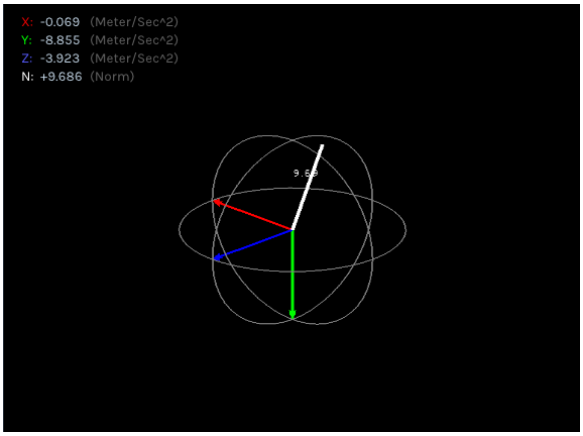

Figure 5-1: Accelerometer data from the Realsense camera used to set up the camera.

Chapter 6

Results

- Influence of different session parameters
- Time of recording
- Time of skeleton tracking

Here we present the results of our experiments. We first present the results of the experiments with the different network layouts. We then present the results of the experiments with the different input data. We also present the results of the experiments with the different error metrics. Finally, we present the results of the experiments with the different data augmentation techniques.

Chapter 7

Conclusion

In Conclusion, ...

7.1 Contribution

In the scope of this thesis, we developed FESDData and FESDModel, or Fault estimation for Skeleton detection for data collection and model creation. FESDData is the tool that allows us to record, analyze and populate it with skeleton 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.

FESDModel is the tool that allows us to train and evaluate the model. ... Still needs to be developed

The code of this thesis is available on GitHub[1].

7.1.1 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)*

7.1.2 Developed Model

7.1.3 Possible applications

FESDModel and FESDData in combination offer users a way to collect and evaluate data for human pose estimation for very specific scenarios. For example in the case of SilverFit a number of exercises can be defined and recorded. The data can then be used to train a model that can be used to estimate the errors of the human pose that might arise with the specific exercise. This can then be used to improve the usage of the pose estimation model in the SilverFit application.

7.2 Future work

- More stability in software (it is actually quite stable already but adding tests and ensuring adequate error handling would make it generally usable) (I also have a lot of

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

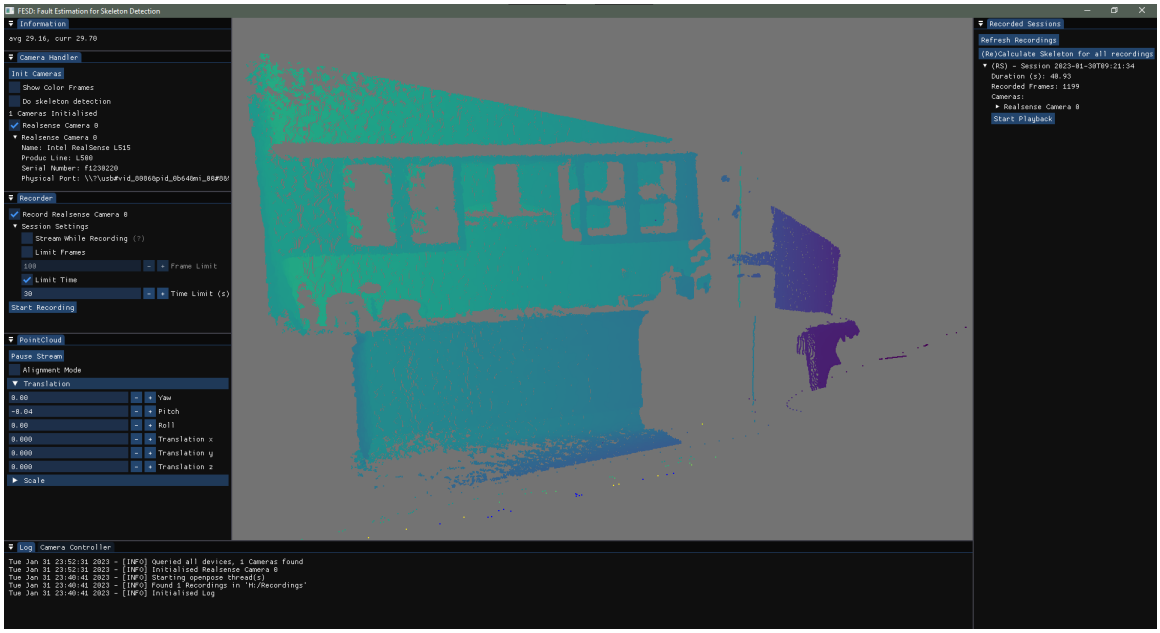

Figure 7-1: A screenshot of the FESD GUI streaming a point cloud. The GUI is used to record and visualize the data, and to playback recordings to validate the data. The GUI is written in C++ using the ImGui framework. The Pointcloud is visualised using OpenGL and a glsl shader.

ideas for this but I will not write them here, maybe ill name some in the report)

- Save everything in a single rosbag file and write a custom rosbag reader
- Multithreaded Streaming and recording (would probably not work since memory would be bottle neck, especially to a single file this would be a problem)
- More data more variation in session parameters
- Different exercises
- Joint Reconstruction
- Attempt to fix detected error
- Import model using ONNX [**?**]
- Use trained model in SilverFit
- Use trained model in HPE sdk
- Use data collector to formulate a dataset for exercise specific HPE
- Train model with different HPE algorithms and use the model to decide when to use which

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] Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. `https://github.com/onnx/onnx`, 2019.

[4] 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.

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

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

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

[8] 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.

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

[10] 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.

[11] Leonhard Euler. Formulae generales pro translatione quacunque corporum rigidorum. *Novi Commentarii academiae scientiarum Petropolitanae*, pages 189–207, 1776.

[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] 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.

[20] 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.

[21] 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.

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

[23] 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.

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

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

[26] 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.

[27] 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.

[28] 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.

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

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

[31] 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.

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