

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

List of Figures

List of Tables

Listings

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 SilverFit¹. 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 ?? 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 ?? Related Work.

¹<https://www.silverfit.com/en/>

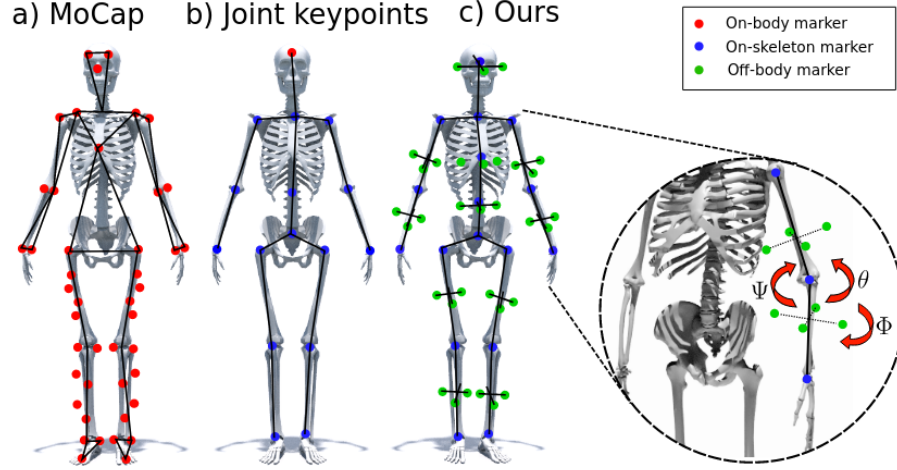


Figure 1-1: Example of a human pose captured with different methods. (a) A captured skeleton using MoCap which we do not focus on in this report. (b) A traditional human skeleton representation. (c) A pose representation that includes the orientation of the joints as well as the bones as presented by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

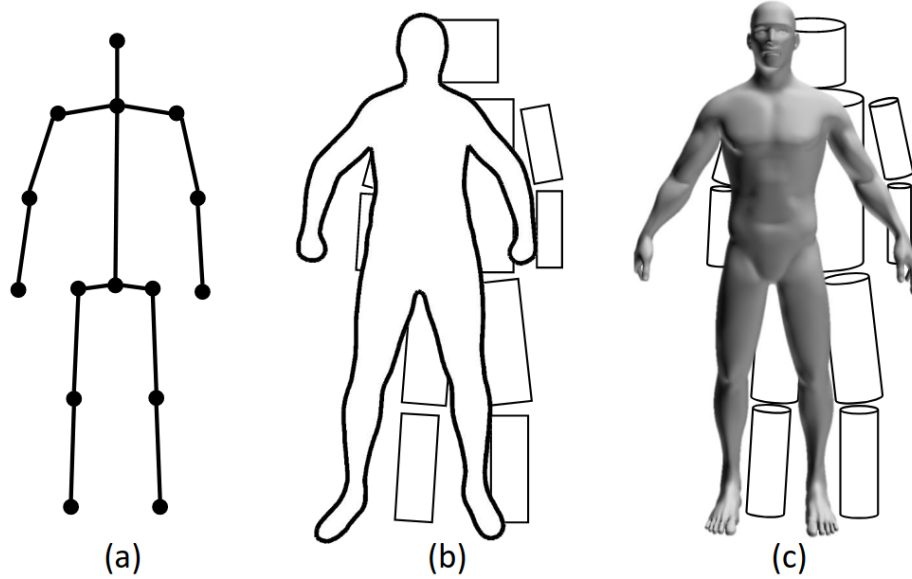


Figure 1-2: Different representations of the human pose. (a) Skeleton representation. (b) Contour representation. (c) 3D volume representation. [?]

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[?]. 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 colored accordingly.

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 human² 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 is 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.

²Or 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 ???. 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.

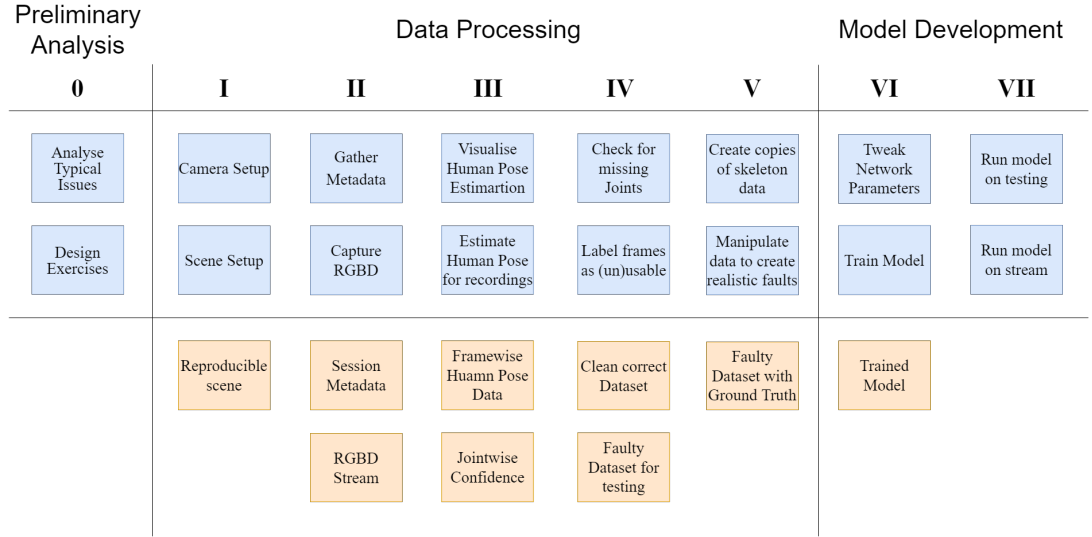


Figure 1-3: The whole Process pipeline with all the steps, which are marked in blue, and the results of each step, which are marked orange. The results of the steps are used as input for the next step. The steps are described in more detail in the following sections. The steps are: **(0)** Preliminary 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 [?]

In [?] 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).

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 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. [?]

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, we will discuss the different approaches that were considered and then we will discuss the approach that was chosen. We then discuss difficulties that were encountered during the data processing process.

As mentioned earlier, we 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 ??, we have already discussed different methods that can be used for human pose estimation. In the scope of this thesis, we will only focus on a single human pose estimator. This has multiple reasons. Firstly, we do not intend to develop a general fault detector. Different pose estimators have different flaws so increasing the pool of pose estimators would presumably create a more general fault detector but less accurate fault detector. Some faults might be generally applicable while others are more specific to a specific estimator. This is especially true if a pose estimator uses a different modality, e.g., OpenPose solely uses RGB data, whereas the human pose estimator proposed by C. Zimmermann et.al utilises RGBD data[?, ?]. 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 Inc¹. 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*[?]. The method uses random forests to estimate the human pose. However, more recently NuiTrack has switched to Deep Learning with Convolutional Neural Networks, which is becoming more and more the industry standard for computer vision tasks such as human pose estimation.

¹<https://nuitrack.com/>

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 discrepancies, 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


With the data prepared and the data layout known, we can start to build a model to predict the errors in the data. As is common practice in computer vision tasks, we use a convolutional neural network to predict the error type of the individual joints. The input of the network are the individual datastreams, i.e. a stream of RGB data, a stream of depth data, and a stream of joint data. The output of the network is a list of error labels for each joint. The network is trained to predict the error labels for each joint. The error labels are the same as the error labels used in the data labeling. The error labels are explained in section ??.

The different modalities are combined in the final three fully connected layers. The model architecture is shown in figure ??.

4.2 Data preparation

In previous sections we have explained the structure of the data and how the data is labeled. In this section we will explain how the data is prepared for training. The data preparation consists of two parts, data augmentation and data splitting.

After the data has been split and augmented, the data is stored in tensors of a fixed size. If an image is too small we apply a bilinear interpolation to resize the image. If an



figures/model_architecture.png

Figure 4-1: Model architecture

image is too large we crop the side. The images are resized to 300×300 pixels. The depth images are resized in the same way and area.

4.2.1 Data augmentation

To ensure that the model is robust to different variations in the data, the data is augmented. We apply three different forms of augmentation to the data. The first form of augmentation is flipping the data. The RGB image, the depth image and the relative joint coordinates are flipped horizontally. Furthermore, we apply random and non-random cropping to the visual data. We ensure that none of the joints are outside the image. We also apply random padding when the image is cropped. The final augmentation is gaussian noise. We apply gaussian noise to the RGB image and the depth image.

4.2.2 Data splitting

The data is split into three parts. The first part is the training data. The training data is used to train the model. The second part is the validation data. We split the training and the validation data vertically, i.e. we split the dataset along frames and not recording sessions. This is done to ensure that the validation data is comparable to the training data.

The validation data is used to validate the model during training. The third part is the test data. The test data is used to test the model after training. The data is split into training data, validation data, and test data using the scikit-learn function `train_test_split`. The data is split into training data and validation data with a ratio of 0.7 to 0.3. The validation data is then split into validation data and test data with a ratio of 0.5 to 0.5. The data is split in this way to ensure that the validation data is always of the same size as the test data. This is done to ensure that the validation data is not biased towards certain classes. The validation data is not biased towards certain classes because the validation data is the same size as the test data.

T

4.3 Model training

4.4 Model evaluation

Finally, we evaluate our model by calculating different error metrics such as the root mean squared error, and the cross entropy loss. 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.

$$\frac{y}{x + y + z} * 90^\circ = \text{Angle in degree}$$

E.g.:

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

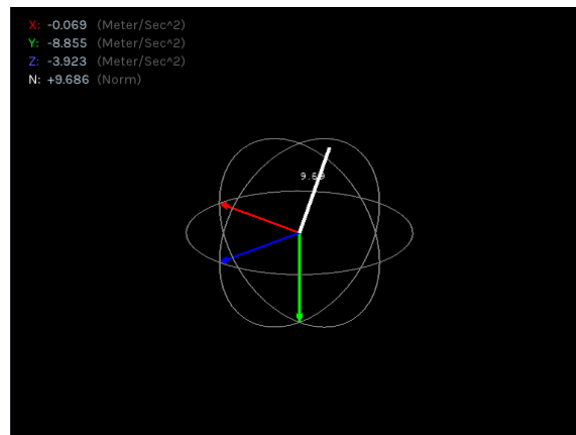


Figure 5-1: Accelerometer data from the Realsense camera used to set up the camera.

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 GitHub¹.

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>

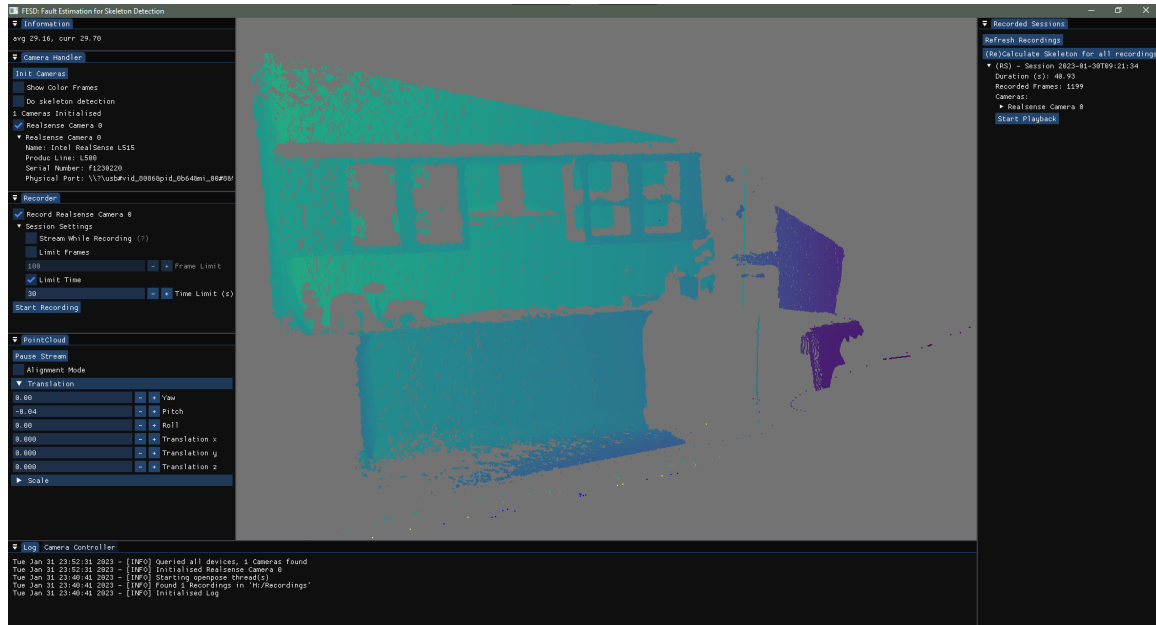


Figure 7-1: A screenshot of the FESD GUI streaming a point cloud. The GUI is used to record and visualize the data, and to playback recordings to validate the data. The GUI is written in C++ using the ImGui framework. The Pointcloud is visualised using OpenGL and a glsl shader.

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