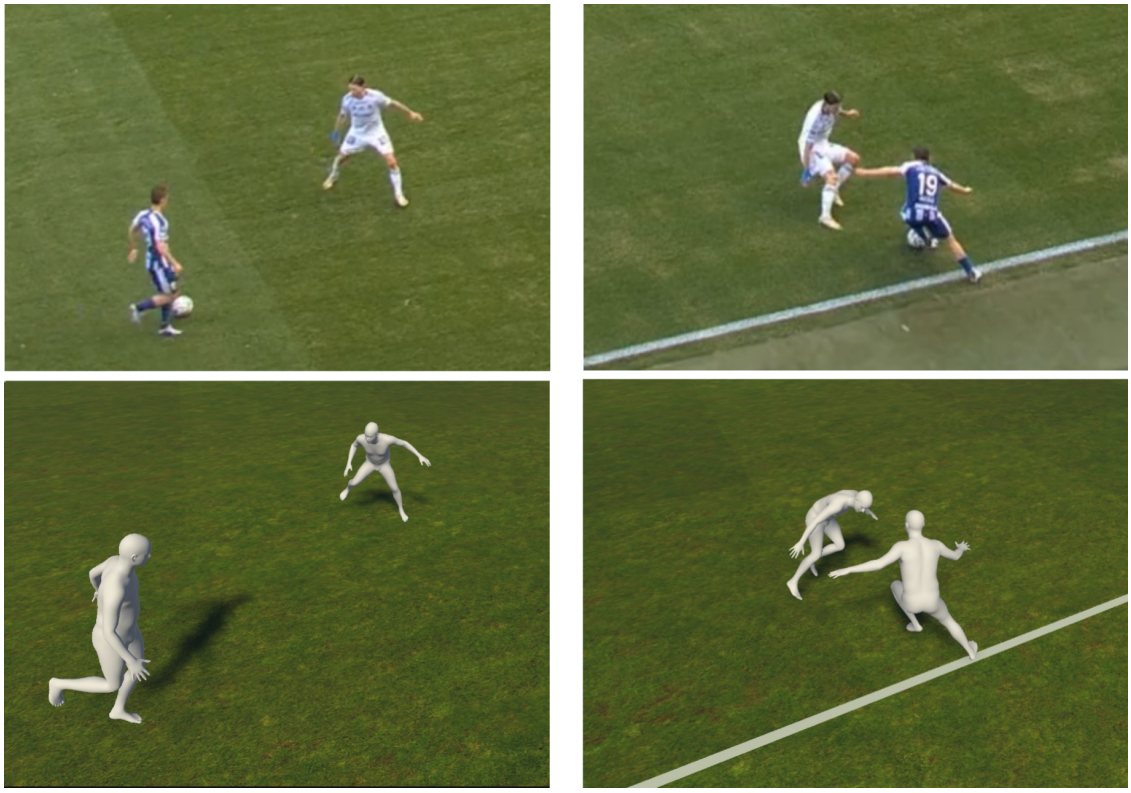
CHALMERS
UNIVERSITY OF TECHNOLOGY



3D Pose Estimation of Football Players

Master's thesis in Data Science and AI

JOAKIM OSTERMAN
OLOF SJÖGREN

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024

Master's thesis 2024

3D Pose Estimation of Football Players

JOAKIM OSTERMAN
OLOF SJÖGREN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

3D Pose Estimation of Football Players
JOAKIM OSTERMAN
OLOF SJÖGREN

© JOAKIM OSTERMAN, 2024.
© OLOF SJÖGREN, 2024.

Supervisor and examiner: Lennart Svensson, Department of Electrical Engineering
Supervisor: Anders Sjöberg, Fraunhofer-Chalmers Centre

Master's Thesis 2024
Department of Electrical Engineering
Division of Signal Processing and Biomedical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Comparison of real and estimated player poses and positions. The top row shows real players on the field, while the bottom row displays corresponding 3D model representations.

Typeset in LaTeX
Gothenburg, Sweden 2024

3D Pose Estimation of Football Players
Joakim Osterman
Olof Sjögren
Department of Electrical Engineering
Chalmers University of Technology

Abstract

In the context of football analytics, video recordings of matches play a crucial role in post-game analysis. However, videos are inherently limited because they only allow viewers to follow the match from the camera's perspective. This thesis is part of a larger project aimed at creating 3D representations of football matches from video, thus enabling users to view the game from anywhere inside the virtual 3D environment. The larger project consists of three parts. This thesis focuses on estimating the camera parameters, as well as the 3D poses and locations of the players in the video. The other two projects focus on player tracking and player texture generation.

A pipeline consisting of camera calibration and pose estimation is proposed, taking video recordings and bounding box annotations as input and predicting camera parameters as well as the players' 3D poses and locations. For camera calibration, a model specifically tailored for cameras viewing football fields is used. The results indicate accurately predicted positions and viewing angles for the estimated camera. Pose estimation is performed using a pre-trained model and results in visually accurate projections, although perspective ambiguities are present when the 3D poses are viewed from different angles. The main approach for positioning players was to detect when players touched the ground and interpolate the positions for ambiguous frames. The results are promising, but noise in the depth estimations occurs due to perspective ambiguities. Subsequently, an optional optimization of poses and positions using multi-view triangulation is also presented, showing possibilities for further refinement to ensure realistic and consistent human poses. Future work on pose and location optimization could yield a pseudo-truth dataset for further enhancements to improve overall poses and positions from strictly monocular video.

Keywords: 3D Human Pose Estimation, Pose estimation, visual transformers, deep machine learning, camera calibration, depth estimation, multi-view optimization.

Acknowledgements

We would like to thank our examiner and supervisor, Lennart Svensson, for his valuable ideas, insights, and guidance. We would also like to thank our supervisor Anders Sjöberg at FCC for his support and helpful contributions. Additionally, we thank the other participants of the overarching project, Filip Anjou & Albin Ekström, and Hugo Ganelius & Jhanzaib Humayun. We also want to express our gratitude towards Bastiand Wandt for his useful feedback and ideas.

Furthermore, we also want to thank FCC for their contributions to this work. Lastly, we are thankful to IFK Göteborg for providing us with football videos and valuable perspectives on coaching feedback and potential use cases of a VR application.

Joakim Osterman, Gothenburg, 2024-06-12

Olof Sjögren, Gothenburg, 2024-06-12

Contents

List of Figures

List of Tables

1

Introduction

Video analysis is a frequently used tool for post-match analysis of football matches [1], [2]. Professional football teams record the matches they play and use these recordings in post-match analysis to coach players and improve gameplay. However, traditional video analysis is limited by the camera's fixed perspective, which often does not align with the players' viewpoints on the field. This makes it hard for players to translate insights from the analysis to practical gameplay improvements since the camera's perspective differs significantly from what the players see on the field [3]. Players and coaches desire the ability to view the match from any arbitrary viewpoint such as from the players' perspectives during the match. This ultimately requires a 3D representation of the football match which necessitates estimating the poses and locations of players on the football field throughout the match.

This project is part of a larger project consisting of three parts. It aims to implement a system that converts videos of football matches into 3D-rendered *Virtual Reality (VR)* environments, where the matches can be replayed and re-enacted. The three sub-projects are intertwined and subsequently build upon each other's results. They consist of:

1. identifying and tracking the players and football throughout the video of the football matches [4],
2. *(this project)* estimating the poses and locations of the tracked football players in 3D,
3. integrating the estimations in a VR environment as well as predicting and rendering player textures [5].

The project is done in collaboration with the professional football team *IFK Göteborg* [6]. The team recognizes great potential in utilizing these videos in a VR environment for post-match analysis to coach players [3].

As stated, this project focuses on the pose and location estimation of football players in 3D, also known as *Human Pose Estimation (HPE)*, which is an active research area within the field of computer vision [7]. The task of pose estimation from images is inherently complex. It requires the capability not only to perceive images but also to identify human subjects accurately, and estimate their specific poses.

One significant challenge arises when the input videos are of low resolution, as is often the case with recordings of football matches. The reduced clarity in low-

resolution footage makes it difficult to identify joints and body parts. This is particularly problematic in a dynamic environment like a football match, where players are often moving and interacting. Moreover, the problem becomes increasingly difficult due to issues like occlusion, where another object covers parts of the subject being assessed. This complexity makes pose estimation a particularly challenging problem, one that traditional computational methods struggle to address effectively. In this context, deep machine learning approaches have emerged as a particularly powerful tool given their ability to process the complex nature of images [7].

In recent years, extensive research has been conducted developing deep learning models for *monocular* HPE. Monocular in this context refers to the utilization of only one camera angle. One such model developed for monocular HPE is the model *HMR2.0*, proposed by Goel et al. [8]. However, the model's estimated poses of football players are often inaccurately displayed in 3D due to projective ambiguities. These ambiguities are not noticeable from the camera's perspective, i.e., the poses appear to be accurately estimated as seen from the camera. But when viewing these poses from different angles in 3D these ambiguities become clear and manifest as unrealistically tilting poses. Moreover, HMR2.0 is not designed to estimate the locations of poses in a common world coordinate system and does not capture the poses' 3D locations relative to each other.

There exist models capable of performing HPE in a global coordinate system; one such model was introduced by Shin et al. [9]. However, it estimates the 3D locations separately for every subject, i.e., the model cannot capture the relation between different poses' 3D locations in a common world coordinate system. Furthermore, it also suffers from inaccuracies during pose estimation in a football setting due to projective ambiguities, similar to HMR2.0. The estimated poses appear accurate from the camera's perspective but are tilted when viewed at other angles in 3D.

Both of these aforementioned models only utilize one camera angle during inference. An additional camera angle would be beneficial to mitigate these projective ambiguities. There exists research that exploits multi-view images to train a monocular 3D HPE model. Wandt et al. [10] introduced a methodology exploiting "multi-view self-supervision". The model is trained on multi-view images but is designed for single-image 3D pose estimation. It predicts both 3D poses and camera rotations, enabling the projection of these poses into another camera view and facilitating the calculation of reprojection errors. By utilizing multiple camera views during training the model learns to accurately represent poses in 3D from different viewing angles, mitigating the problem of tilting poses. However, this model is built upon an initial 2D joint detector to predict a 3D skeletal pose, unlike a 3D mesh prediction from bounding boxes. Also, it is not targeted at performing HPE of several subjects in an image and it cannot capture the relations between poses' 3D locations. Nevertheless, utilizing multi-view images to better estimate 3D poses is a promising approach for optimizing monocularly estimated poses.

Despite these outlined models, no single methodology currently addresses all challenges inherent in 3D HPE for football players in a VR application. This thesis explores strategies to build upon existing methods for pose and location estimation

within a common 3D world coordinate system in the context of football. Videos of IFK football matches are used to explore these strategies. These videos are often captured from multiple angles, which enables an optional approach for optimizing the estimated poses and locations using two camera angles. Finally, calibration of the cameras is also performed to position the players.

1.1 Objective

This project aims to create a system for HPE in the specific context of football. More specifically, the objective is to create an approach for:

- performing HPE in 3D of low-resolution players in football videos,
- determining the 3D location of players on the football field,
- optimizing estimated poses and locations using two different camera angles of the same player.

1.2 Scope

Several HPE methods are designed to first identify human subjects and their respective *bounding boxes*, before performing pose estimation. A bounding box is a rectangular box that tightly encloses a specific subject in the image and is typically used for tracking or detection purposes in computer vision applications. These methods then perform HPE on each identified bounding box in each frame. This problem of extracting the bounding boxes and tracking them over time in videos are tasks assumed to be solved by Project 1 and are therefore not covered by this project. So, this project assumes annotated videos with tracking of football players are readily available.

This project utilizes an existing pre-trained deep learning model capable of performing 3D HPE. Therefore, no training of the implemented HPE model is covered. Additionally, the HPE in this project is performed in an offline context, i.e., the input is videos of fixed length. This is opposed to an online approach, where a model works in real-time as the video is being recorded. The main difference between these contexts is the availability to access data in future frames in offline settings. In an offline setting, the computational time of the model can be substantially longer as it is not required to output a live feed of the processed video.

Additionally, the videos of football matches provided by IFK use a stationary camera that is fixed throughout the match. Therefore the project is subsequently limited to pose estimation in a context where the camera does not move or change orientation. A frame from one of IFK's videos can be seen in Fig. 1.1. It illustrates how much of the football field the camera capturing the videos typically sees.

Figure 1.1: Example of the camera's perspective in a video of a football match provided by IFK.

Figure 1.2: An example of how the resolution of players might vary depending on their distance from the camera used to record the video. The player to the right is far away from the camera and the enclosing bounding box is subsequently smaller, thus enclosing fewer pixels. This makes the player significantly more pixelated than the player on the left which is closer to the camera, i.e., the bounding box is larger and contains more pixels.

Moreover, the cameras recording the matches generally yield low-resolution videos for each player, although some players might be higher resolution depending on how close they are to the cameras. This is exemplified in Fig. 1.2. It shows how an image of a player farther away from the camera is noticeably lower resolution than a player closer to the camera.

It should also be noted that IFK is a professional football team comprised of adult male football players. Consequently, the project specifically focuses on accurately capturing the poses and movements of adult male players. Pose estimation of other demographics, such as children, is not covered in this project.

To summarize, this project is specifically situated for the pose and location estimation of adult male professional football players. It utilizes pre-trained deep-learning models for pose estimation using annotated tracking of players across frames which are assumed to be pre-processed and available. The pose estimation is performed in an offline setting, allowing for computational processing without real-time constraints. The videos of IFK matches have been captured by static cameras and are of fixed length with varying resolutions of the captured players.

2

Theory

This chapter covers the relevant theory used in the report. Camera geometry and calibration theory will be detailed, followed by theory concerning pose representation. Furthermore, background on the transformer architecture and how it is applied in image contexts is provided before the relevant HPE model used in the project is described. Lastly, the necessary tools for animating estimated poses in a 3D environment are presented.

2.1 Camera Geometry and Calibration

This section presents relevant theory for the mathematical camera model used throughout the project. It also covers the details regarding the pre-trained model used for camera calibration.

2.1.1 The Pinhole Camera

The pinhole camera model is frequently used as a mathematical representation of a camera [11]. The model is visualized in Fig. 2.1. It consists of a 3D coordinate system $(X_c, Y_c, Z_c)^\top$ with the camera located at the origin (the camera center). This system (with the camera at the origin) is the *camera coordinate system*, and the camera is typically modelled as looking down the Z_c-axis, i.e., its principal axis. The image plane is defined by $(x, y, f)^\top$, typically with $f = 1$, and can be thought of as the 2D representation of what the camera sees in the 3D world. Any point on the image plane $(u, v)^\top$ is the intersection of some viewing ray defined as the line passing through the camera, image plane, and originating 3D point $\boldsymbol{p} = (x, y, z)^\top$. Note that any 3D point along this viewing ray results in the same projected 2D point. The 2D representation of the projected 3D point on the image plane is obtained by dropping the Z-dimension. The perspective projection π of a 3D point $\boldsymbol{p}$ along its viewing ray to the image plane of the camera is defined as:

$$\pi(\boldsymbol{p}) = \left(\frac{x}{z}, \frac{y}{z}\right)^\top. \tag{2.1}$$

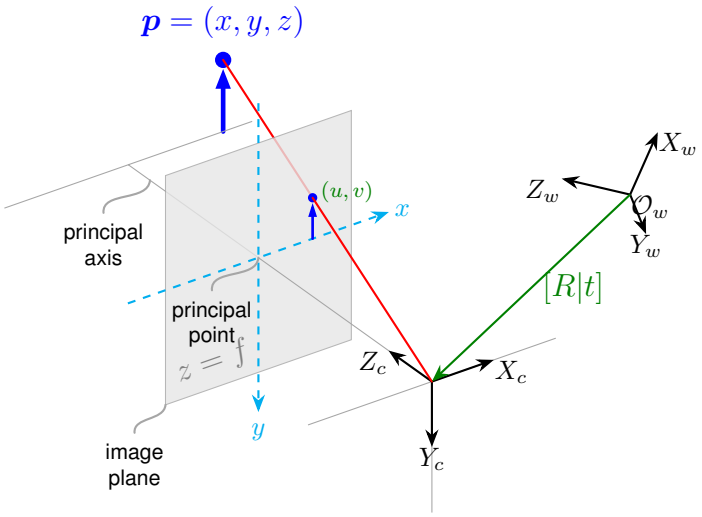


Figure 2.1: The Pinhole Camera model.

This models how the 3D points in the world are captured in a 2D image. However, it should be noted that the reverse process of going from a 2D image to a 3D representation is inherently more ambiguous. This is due to the dimensionality reduction that occurs when projecting a 3D point onto a 2D image. In other words, it is not directly possible to determine which 3D point resulted in the projected 2D point; as previously mentioned, any point along the viewing ray projects to this 2D point.

A *world coordinate system* $(X_w, Y_w, Z_w)^\top$ is frequently used to capture the relative position and viewing angles of multiple cameras in relation to each other. So, each camera has its own coordinate system, but they are all placed in a common world coordinate system. Subsequently, each camera has its own position and viewing angle measured from this world coordinate system. To translate between the world coordinate system and a camera's coordinate system, the camera's translation vector $\boldsymbol{t}$ and rotation matrix R are used:

- $\boldsymbol{t} \in \mathbb{R}^3$ is a translation vector and defines the origin of the world coordinate system in relation to the origin of the camera coordinate system.
- $R \in \mathbb{R}^{3\times3}$ is a rotational matrix and defines the camera's viewing angle, i.e., its viewing direction according to the world coordinate system.

These are called the *extrinsic camera parameters*. The calibrated camera matrix is defined as:

$$(R \mid \boldsymbol{t}) = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix}. \tag{2.2}$$

It can be used to translate a 3D point $\boldsymbol{p}$ (defined in the world coordinate system) into the camera's coordinate system by $(R \mid \boldsymbol{t})\,(\boldsymbol{p}, 1)^\top$, first translating the coordinate to a homogeneous coordinate by setting the w-dimension to 1.

Moreover, the *intrinsic camera parameters* define the internal characteristics of the camera that affect how it captures an image. These parameters include the focal length, the principal point, and any skew between the image axes. They are encapsulated in the intrinsic camera matrix, denoted by K:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.3}$$

where:

- f_x and f_y are the focal lengths along the x and y axes, respectively,
- s is the skew coefficient, which is typically zero for most cameras,
- c_x and c_y are the coordinates of the principal point.

So, the full pinhole camera model is represented by:

$$P = K\,(R \mid \boldsymbol{t})\,. \tag{2.4}$$

2.1.2 SoccerSegCal for Camera Calibration

SoccerSegCal is a "Soccer Pitch Segmentation and Camera Calibration" model developed by Spiideo and designed for the camera calibration competition held by SoccerNet in 2023 [12], [13]. It performs camera calibration in a two-step fashion. First, it performs a pixel-level segmentation of the input image followed by an optimizer that estimates the camera parameters from the segmented image.

The first step is performed by a DeepLabV3 CNN [14], which segments pixels into six different classes. These classes are predefined and separated by the different areas and line markings on a football field (e.g., center circle, penalty area, etc.). The second step is to utilize this pixel-level segmentation with a 3D Soft Rasterizer to render a synthetic 3D field with a set of camera parameters. This step also involves projecting the synthetic 3D field into 2D space to simulate how the field appears from the set of camera parameters. These camera parameters are then iteratively optimized to minimize the difference between the real segmented image and the rendered image. By doing this, the model estimates the optimal camera parameters that align the virtual field with the segmented image. This optimization uses the local optimizer AdamW [15].

The model was trained and tested on the annotated dataset for camera calibration provided by SoccerNet. The dataset for training contains 20 028 images with annotated classes for all football lines and goal posts. The test set is comprised of 2 104 images annotated in the same manner.

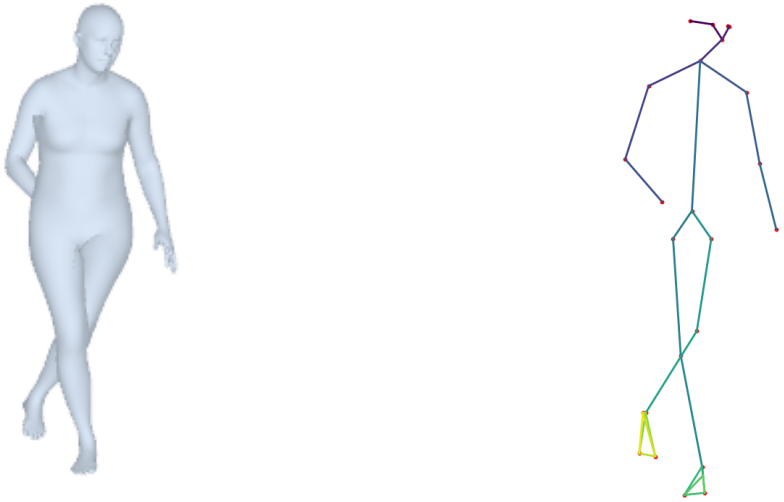


Figure 2.2: Examples of human pose representations; the left pose is a 3D mesh-grid representation (the SMPL model) and the right pose is a skeletal representation.

2.2 Pose Representation

There are several approaches for quantitatively representing a human pose [16]. Two frequently used approaches consist of representing the human pose as either a 3D mesh-grid model or a skeletal-based model, exemplified in Fig. 2.2.

2.2.1 Skeletal Pose Representation

The skeletal model consists of a stick-figure representation; it is comprised of several limbs connected by keypoints in the form of a human body. Each keypoint typically represents a joint such as a knee, elbow, or hip. The actual amount of keypoints may vary between representations with some containing more than others, thus giving more detail. For instance, a hand may be represented as a single keypoint in the middle of the palm, or by multiple keypoints with one for each of the joints in the fingers of the hand.

Ultimately, the skeletal representation is useful as it can give a lightweight yet concrete representation of the overall pose; a joint can be succinctly defined as either a 3D point or a 2D point, and the limb is defined as the line between two joints. This facilitates comparison between two skeletal poses as one can measure the distance between the corresponding joints. However, the skeletal model fails to capture other nuances of the human body such as the skin deformations, muscle definition, and volumetric information, which are crucial for a more complete and realistic representation.

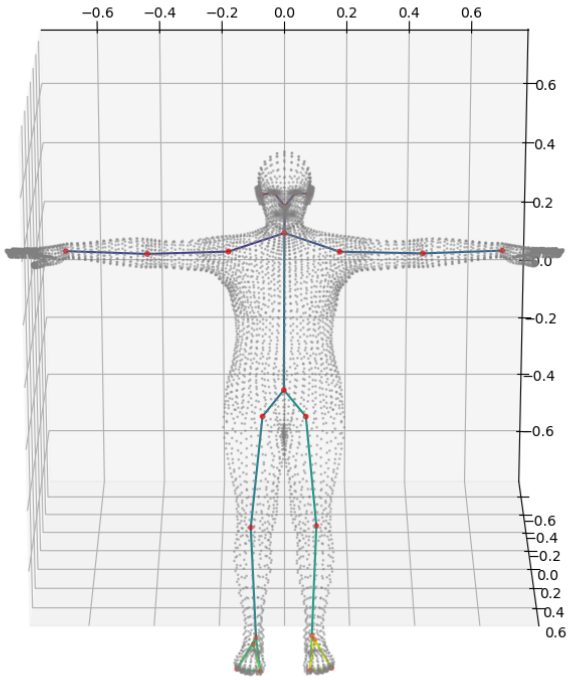


Figure 2.3: The mesh representation of the SMPL model along with its joints and skeletal structure.

2.2.2 Skinned Multi-Person Linear Model (SMPL)

The other popular way of representing a human is using a deformable 3D mesh modeled in the shape of a human. A widely used model is the *Skinned Multi-Person Linear Model (SMPL)* [17], which is a vertex-based model whose pose and deformation are controlled by a set of parameters. The mesh $M \in \mathbb{R}^{3\times 6890}$ consists of 6890 3D vertices modeled in the form of a human reflecting the pose of an underlying skeletal model which in turn consists of 23 different 3D joints $J \in \mathbb{R}^{3\times 23}$. The skeletal model is a kinematic tree; the location of each part is determined by the relative rotation to its parent in the tree, with the hip keypoint being the root. The SMPL model is visualized in Fig. 2.3.

The deformation of the mesh is controlled by the different parameters $(\boldsymbol{\theta}_b, \boldsymbol{\theta}_g, \boldsymbol{\beta})$:

- $\boldsymbol{\theta}_b \in \mathbb{R}^{23\times 3}$ controls the axis-angle representation of the relative rotation for each of the 23 joints. More concisely, it controls the pose of the body by defining how each part of the limbs in the body are rotated relative to each other.
- $\boldsymbol{\theta}_g \in \mathbb{R}^3$ controls the global orientation of the model, also represented as an axis-angle rotation.
- $\boldsymbol{\beta} \in \mathbb{R}^{10}$ controls the deformation and overall shape of the model, adjusting properties such as height and width, represented as 10 scalar values.

So, given the set of parameters $(\boldsymbol{\theta}_b, \boldsymbol{\theta}_g, \boldsymbol{\beta})$, the model produces the mesh M. There exists a linear mapping consisting of weights $W \in \mathbb{R}^{6890\times 23}$ supplied by SMPL from which the joints J can be extracted, i.e., $J = MW$.

SMPL is able to more accurately capture the pose of a human being as it models the skin deformations, muscle dynamics, and overall body volume, aspects that go beyond the skeletal representation's focus on joints and limbs alone. Moreover, the ability to also retrieve a skeletal representation of an SMPL pose facilitates the comparison between different poses. This is crucial for the evaluation and comparison of poses in contexts such as pose estimation.

2.3 Transformers

Transformers are a type of deep learning architecture originally developed for *Natural Language Processing (NLP)* tasks and was published in the paper "Attention is all you need" [18]. The architecture was designed to be more parallelizable and to require significantly less training time compared to earlier models like *Recurrent Neural Networks (RNNs)*. Since its release, transformers have become widely used for applications beyond NLP tasks such as HPE.

Transformer architectures are built upon an encoder-decoder structure. The purpose of the encoder is to map the input sequence to some sequence of continuous representation which can be fed into the decoder. The decoder aims to generate an output from the given input encoder sequence. To pass input information to the encoder, the input must first be converted into tokens, usually done by some tokenizer.

A token is a basic unit of data, often represented as a vector. For NLP tasks the tokenizer converts text into tokens. An embedding layer then converts these input tokens and their respective positions into continuous representations known as embeddings, represented as vectors. These embeddings are then fed into the transformer model. The original transformer architecture is visualized in Fig. 2.4.

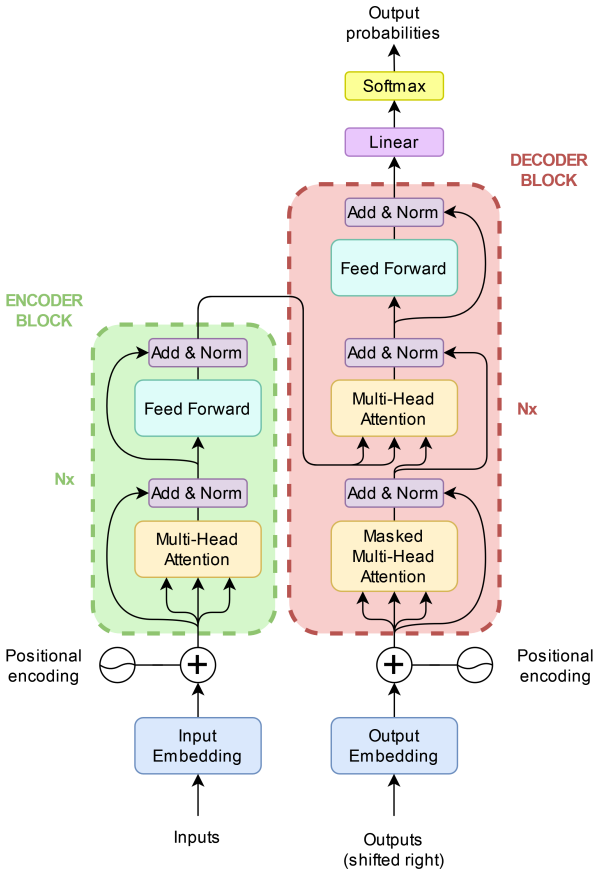


Figure 2.4: Visualization of the original transformer network architecture [18]. The green block to the left is the encoder, and the red block to the right is the decoder.

2.3.1 The Encoder Block

The encoder block combines a multi-head attention mechanism, layer normalizations, and a feed-forward neural network to transform input sequences into some high-dimensional contextual representation. The originally proposed encoder is built on a stack of six identical layers; each layer is composed of two main sub-layers [18].

The first is the multi-head self-attention mechanism followed by the second sub-layer of a fully connected feed-forward network. The multi-head self-attention is formed by several attention layers running in parallel, referred to as heads. This mechanism allows the model to capture long-range and complex dependencies between different parts of the input sequence. The attention function computes an output by mapping a query with a set of key-value pairs, generating a weighted sum of values where the weights reflect the compatibility between the query and the corresponding key. The input queries and keys are of dimension d_k and the values are of dimension d_v. A dot product is computed and scaled, and a softmax function is applied to obtain weights. This function is computed on a set of queries, keys, and values that are packed into matrices Q, K, and V. The output matrix is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2.5}$$

The scaling factor of $\frac{1}{\sqrt{d_k}}$ is important to counteract gradients becoming extremely small in certain regions. The queries, keys, and values are linearly projected h times with distinct learnable projections in the multi-head attention. An attention function runs in parallel on each of these projections, producing output values of dimension d_v, which are also concatenated. So, the multi-head self-attention output is computed as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O, \tag{2.6}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \tag{2.7}$$

where the matrices W_i^Q, W_i^K, and W_i^V are the projection matrices for the $head_i$, and W^O is the output projection matrix for the entire multi-head attention layer.

Before each of the two sub-layers, residual connections [19] are used around the respective layer, followed by a layer normalization. These residual connections mitigate problems of vanishing gradients by enabling gradients to propagate through the network more easily. The generated output for a sub-layer can be expressed as $\text{LayerNorm}(\boldsymbol{x} + \text{Sublayer}(\boldsymbol{x}))$, where $\text{Sublayer}(\boldsymbol{x})$ is the specific function output by that sub-layer.

The fully connected feed-forward sub-layers are employed individually for each position. They consist of two linear transformations with a *Rectified Linear Unit (ReLU)* activation in between and these transformations are the same across different positions but with different parameters from layer to layer. The feed-forward network output is expressed as:

$$\text{FFN}(\boldsymbol{x}) = \max(0, \boldsymbol{x}W_1 + \boldsymbol{b}_1)W_2 + \boldsymbol{b}_2. \tag{2.8}$$

2.3.2 The Decoder Block

The decoder block is almost identical to the encoding block. The originally proposed decoder block is also composed of 6 identical layers stacked together but with the addition of an extra sub-layer compared to the encoder. The first sub-layer of the decoder is modified such that it only attends to the previous words, while the encoder is designed to attend to all the words of the input sequence. Therefore, a prediction for a token at position i is only dependent on the known outputs for the previous tokens in the sequence. This is achieved through a masking combined with an offset of one position. In other words, this makes the first sub-layer of the decoder unidirectional in contrast to the bidirectional encoder.

The second sub-layer of the decoder is the additionally added multi-head attention layer, compared to the encoder. This layer has been implemented to receive its keys and values from the output of the encoder while the queries are received from its previous decoder sub-layer. This process is often referred to as *cross-attention* and it allows the decoder to gain better contextual understanding. This is because the cross-attention allows attention to all of the tokens in the input sequence. This layer is followed by a fully connected feed-forward network, similar to the one in the encoder block. Like the encoder, there are residual connections followed by a normalization layer around every sub-layer inside the decoder block.

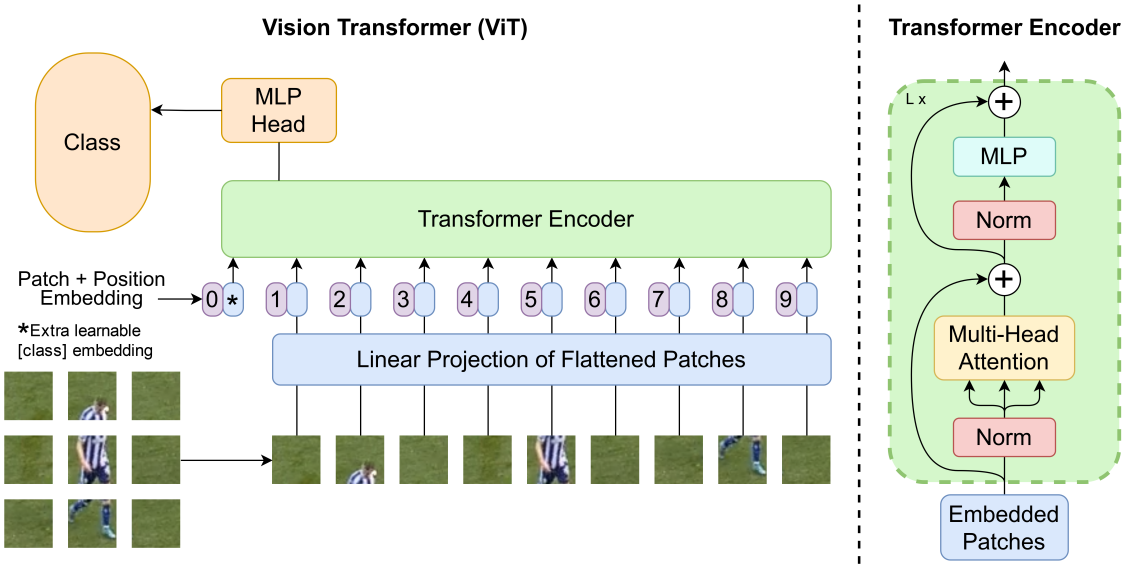


Figure 2.5: Visualization of the vision transformer (ViT) network architecture to the left. To the right is the illustration of the transformer encoder used by the ViT.

2.4 Vision Transformer (ViT)

Vision transformers were first introduced by Dosovitskiy et al. in 2020 [20] as an adaptation of the transformer architecture for computer vision. It was introduced as an encoder-only transformer; no decoder block was included. In short, it functions by splitting an input image into squared patches to be used as its sequence tokens.

The input image is of size $\mathbb{R}^{H\times W\times C}$ where H, W are the height and width, and C is the number of channels with the split patches of type $\mathbb{R}^{N\times(P^2\times C)}$. The resolution of each patched image is of $P \times P$ where $N = HW/P^2$ is the number of patches the input image is split into. Each patch is linearly embedded to obtain a patch embedding vector, representing the information in that image patch. Additionally, the position of every patch is also transformed into a vector by a positional embedding. These two vectors are added and the resulting sequence of vectors is passed into a standard transformer encoder. An overview of the ViT model and its transformer encoder is illustrated in Fig. 2.5.

A noteworthy difference in Fig. 2.5 is that the layer normalization has been shifted from the original transformer by Vaswani et al. [18]. It was concluded that transformers with layer normalization inside the residual blocks, as opposed to outside, enable more efficient training and faster convergence speed of transformers [21]. Another important part of the ViT is the classification token (CLS token), marked with an asterisk above. This token acts as an extra learnable token to the input sequence. It represents the information in the input image and can be utilized for classification tasks.

2.5 Pose Estimation

Estimating poses from images becomes more complex when there is an unknown number of subjects present. There are generally two different approaches when solving HPE for an unknown number of persons in the given input image: *bottom-up* and *top-down* methods.

Bottom-up approaches start with an initial detection of every possible joint location followed by a subsequent grouping of these joints into individual poses. Each keypoint detection is typically associated with a corresponding heatmap representing grid areas of pixels indicating confidence levels for the respective keypoint detections.

Top-down approaches on the other hand start by first detecting every person in the image, often as bounding boxes. Then, a pose estimation algorithm is applied independently for each detected person to create pose estimations. The pose estimation is carried out within each respective bounding box, i.e., any information outside that bounding box is disregarded.

2.5.1 HMR2.0

Human Mesh Recovery 2.0 (HMR2.0) is a ViT-based HPE model developed by Goel et. al [8], building upon the previous HPE framework *Human Mesh Recovery (HMR)* developed by Kanazawa et. al [22]. It assumes a top-down HPE approach as it takes the bounding box of a person as input and produces a SMPL pose representation.

Given a 256×192 image of a human subject, the model predicts a set of SMPL parameters ($\boldsymbol{\theta}_b \in \mathbb{R}^{23\times3}, \boldsymbol{\theta}_g \in \mathbb{R}^3, \boldsymbol{\beta} \in \mathbb{R}^{10}$) representing the person's pose and shape in the picture. It also predicts a camera translation $\boldsymbol{t} \in \mathbb{R}^3$ for the camera viewing the image (but no camera rotation as the global orientation of the pose $\boldsymbol{\theta}_g$ is a part of the SMPL parameters).

The model uses the encoder of ViT-H/16 [20] for feature extraction of the image, which consists of 50 transformer layers outputting 16×12 feature embedding vectors of size 1280. The output feature embeddings are then passed to the transformer decoder. It is implemented using a transformer architecture with 6 layers, similar to the original transformer [18]. Each layer includes multi-head self-attention, multi-head cross-attention, a feed-forward network, and layer normalization:

- The model has a hidden dimension of 2048.
- Eight attention heads are used for both self-attention and cross-attention, with each head having a dimension of 64.
- The feed-forward networks within each layer have a hidden dimension of 1024.

The decoder takes a single learnable SMPL token of size 2048 as input and cross-attends to 192 image tokens arranged in a 16×12 grid. Finally, a linear layer is applied to the output token from the transformer decoder and produces the predicted SMPL parameters ($\boldsymbol{\theta}_b, \boldsymbol{\theta}_g, \boldsymbol{\beta}$) and camera translation $\boldsymbol{t}$.

The model was trained using the datasets Human3.6M [23], MPI-INF-3DHP [24], COCO [25], MPII [26], InstaVariety [27], AVA [28], and AI Challenger [29]. Different losses were used depending on the format of the labeled data. When the labels consisted of SMPL parameters, the L2 loss was used for the pose and body parameters $(\boldsymbol{\beta}, \boldsymbol{\theta}_b)$:

$$\mathcal{L}_{\text{SMPL}} = \|\boldsymbol{\beta}_{\text{pred}} - \boldsymbol{\beta}_{\text{true}}\|_2^2 + \|\boldsymbol{\theta}_{b,\text{pred}} - \boldsymbol{\theta}_{b,\text{true}}\|_2^2 , \tag{2.9}$$

where $\boldsymbol{\beta}_{\text{pred}}$ and $\boldsymbol{\beta}_{\text{true}}$ represents the predicted and ground truth body shape parameters, and $\boldsymbol{\theta}_{b,\text{pred}}$ and $\boldsymbol{\theta}_{b,\text{true}}$ represents the predicted and ground truth body pose parameters of the SMPL model.

For 3D joint labels, the L1 loss was used between the true and predicted 3D joints from the SMPL model:

$$\mathcal{L}_{\text{3D}} = \|\boldsymbol{J}_{\text{3D, pred}} - \boldsymbol{J}_{\text{3D, true}}\|_1 , \tag{2.10}$$

where $\boldsymbol{J}_{\text{3D, pred}}$ and $\boldsymbol{J}_{\text{3D, true}}$ are the predicted and ground truth 3D joint locations, respectively.

Finally, when only 2D joint labels were available, the L1 loss was also used after projecting the 3D points onto the image:

$$\mathcal{L}_{\text{2D}} = \|\pi(\boldsymbol{J}_{\text{3D, pred}}) - \boldsymbol{J}_{\text{2D, true}}\|_1 , \tag{2.11}$$

where π is the projection function that maps predicted 3D joint positions $\boldsymbol{J}_{\text{3D, pred}}$ to 2D image coordinates, and $\boldsymbol{J}_{\text{2D, true}}$ are the ground truth 2D joint locations.

Moreover, an adversarial loss was employed by incorporating a discriminator network $\mathcal{D}$, as proposed in the original HMR paper. An overview of the discriminator's architecture is visualized in Fig. 2.6. The discriminator is used to evaluate the plausibility of the predicted poses and it was trained on the same datasets. Specifically, it predicts the likelihood that a given pose is realistic, thereby theoretically penalizing any inhuman or improbable poses. The inputs to the discriminator are the SMPL pose and body shape parameters $(\boldsymbol{\beta}, \boldsymbol{\theta}_b)$. It outputs a vector of probabilities in the range $[0, 1]$ for the joint rotations, shape parameters, and overall pose. The adversarial loss $\mathcal{L}_{\text{adv}}$ is computed by summing the squared differences between each element of this vector and 1, ensuring that the final loss reflects the overall realism of the predicted poses as evaluated by the discriminator. The discriminator loss for the SMPL parameters is subsequently calculated as:

$$\mathcal{L}_{\text{adv}} = \sum_{j=1}^{M} \left(\mathcal{D}_j(\boldsymbol{\beta}, \boldsymbol{\theta}_b) - 1\right)^2 , \tag{2.12}$$

where $M = 25$ is the number of elements in the discriminator output vector, $\boldsymbol{\beta}$ represents the body shape parameters, $\boldsymbol{\theta}_b$ represents the body pose parameters, and $\mathcal{D}_j(\boldsymbol{\beta}, \boldsymbol{\theta}_b)$ is the j-th element of the discriminator output vector.

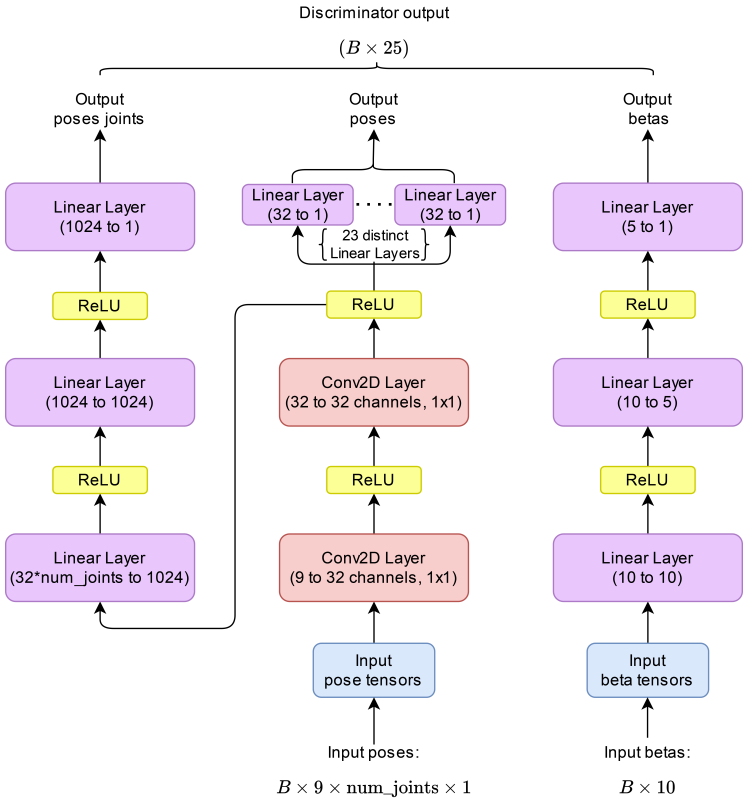


Figure 2.6: Visualization of the discriminator architecture used by HMR2.0.

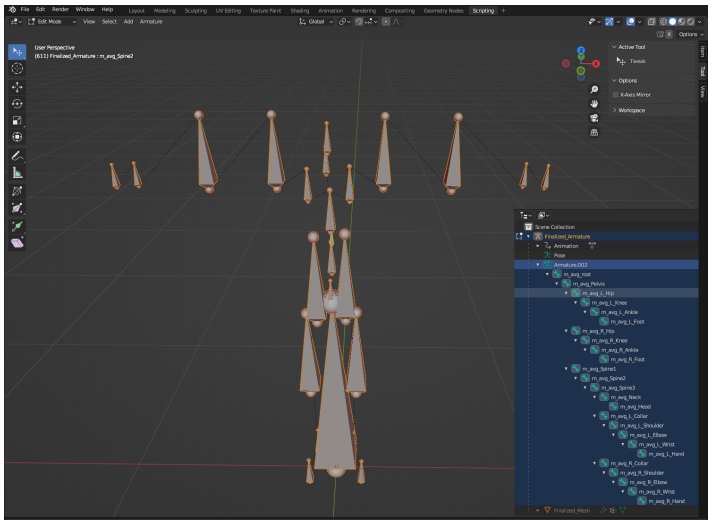
2.6 Blender for Animation Generation

Blender is an open-source software tool designed for 3D applications [30]. It allows for 3D modeling, sculpting, animating, rendering, and more. Additionally, blender supports a Python API, referred to as BPY [31]. This API supports the entire 3D pipeline and allows users to script and automate the steps carried out by Blender.

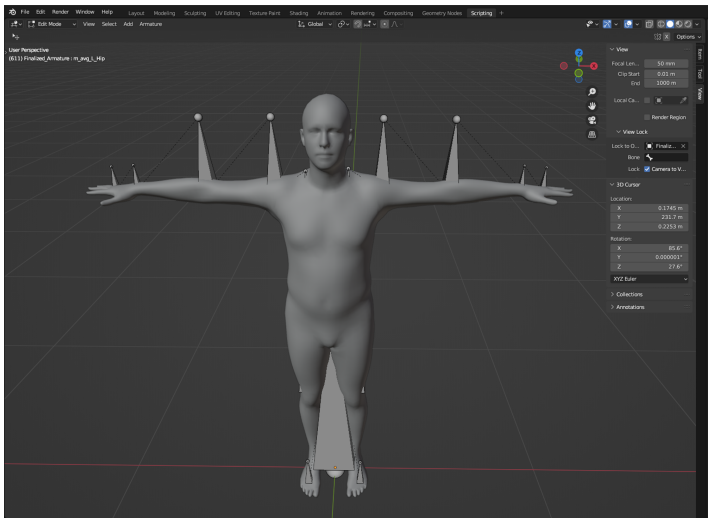
2.6.1 Armature Object

An armature object can be seen as Blender's way of representing a real skeleton. It is structured as a hierarchy of bones that act as children and parents, and the bones often have attachments such as child bones or vertices linked to them. Each armature has an individual root that acts as the parent object to all other bones that are included in the armature. Thus, every bone in the armature lies under the root in the hierarchy.

When bones are moved and rotated their respective attachments will move and deform similarly. This is important when changing the position of the animated pose so that the entire armature moves together. Moreover, animating a pose is done by a *Keyframe insertion*, essentially updating the 3D orientation and position of the armature for the currently selected frame. A SMPL armature and its mesh grid are visualized in Fig. 2.7.



(a) Armature object and all the bones.



(b) Mesh grid attached to the armature.

Figure 2.7: Visualization of (a) SMPL armature and (b) mesh grid.

2.6.2 Quaternions and 3D Rotations

Blender allows users to define and apply rotations in different ways called rotation modes [32]. The different modes are Euler mode, Axis Angle mode, and Quaternion mode. The quaternion mode inside Blender is ideal for interpolating between any pair of orientations, particularly useful when animating armatures over different frames.

Quaternions are a way to represent rotations in 3D space. In contrast to Euler angles for representing 3D rotations, quaternion representations are not susceptible to issues such as *gimbal lock*. Gimbal lock is the loss of one degree of freedom when describing rotations within a multi-dimensional space. Specifically, in 3D space, a gimbal lock occurs when two axes become parallel and aligned, restricting certain 3D rotations, i.e., going from three to two degrees of freedom for rotations.

Quaternions ensure stable and accurate rotational transformations in 3D space. They are composed of a real (scalar) part and an imaginary (vector) part and offer compact means of expressing rotations while being computationally efficient. Despite their effectiveness, visualizing and interpreting quaternions is complex since they exist in a four-dimensional (4D) space. A quaternion is mathematically expressed as:

$$\boldsymbol{q} = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}, \tag{2.13}$$

where $a, b, c, d \in \mathbb{R}$ are coefficients and $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are the base vectors.

Quaternions used for representing rotations in 3D are known as *unit quaternions*, or *versors*, and per definition, they have a magnitude of one. The quaternions inside Blender are of this normalized type. This normalization ensures that the quaternion lies on the unit hypersphere in a 4D space. This provides quaternions with a simple way of interpolating between two different rotation angles in 3D, which is important during 3D animation.

Another interesting property of unit quaternions for 3D rotations is that the negative of a unit quaternion represents the same rotation. So the quaternion $\boldsymbol{q} = \frac{1}{\sqrt{2}} + 0\boldsymbol{i} + \frac{1}{\sqrt{2}}\boldsymbol{j} + 0\boldsymbol{k}$ and $\boldsymbol{q} = -\frac{1}{\sqrt{2}} + 0\boldsymbol{i} + -\frac{1}{\sqrt{2}}\boldsymbol{j} + 0\boldsymbol{k}$ both represent the same 90° rotation around the y-axis. Additionally, this rotation can be expressed as a rotation around a unit vector.

3

Methods

This chapter describes the process of estimating pose and location using the annotated input video recordings. Initially, an overview of the system is presented describing how the different parts of the pose estimation process are connected. Each part of the process is subsequently described in more detail in separate sections. These cover the input data used, the calibration of the camera, the pose estimation, the process of locating the players in a 3D environment, further multi-view pose and location optimization, and finally conversion to a 3D environment.

3.1 System Overview

The system was structured as a pipeline, divided into several sections. It processes annotated video recordings to produce a 3D environment that includes the representations of football players from the video, exported as a single file. The parts of the pipeline are visualized in Fig. 3.1.

As seen in the leftmost part of the figure, the pipeline takes bounding box annotations generated by Project 1 [4] and a video recording as input. The bounding box annotation consists of a single `.csv` file and the video an `.mp4` file. The first video frame is extracted using the video processing tool `ffmpeg` [33] and subsequently fed to the camera calibration model. Only the first frame is required since the camera is static meaning its parameters do not change throughout the video. In parallel to the camera calibration, the pose estimation model processes both the video and the provided bounding box annotations to predict the poses and shapes of players in each video frame, expressed as SMPL parameters.

An optional pose optimization step is proposed as well which optimizes the predicted SMPL poses and 3D location using the calibrated cameras. This step is optional in the sense that it can be skipped and the originally predicted SMPL parameters from the pose estimation model can be used instead. The final step of the pipeline is the Unity conversion step which utilizes the 3D software Blender to render and create a 3D environment of the video recording using the predicted (and possibly optimized) SMPL parameters and camera parameters. Unity is a game engine and is used by Project 3 to construct the VR environment [5].

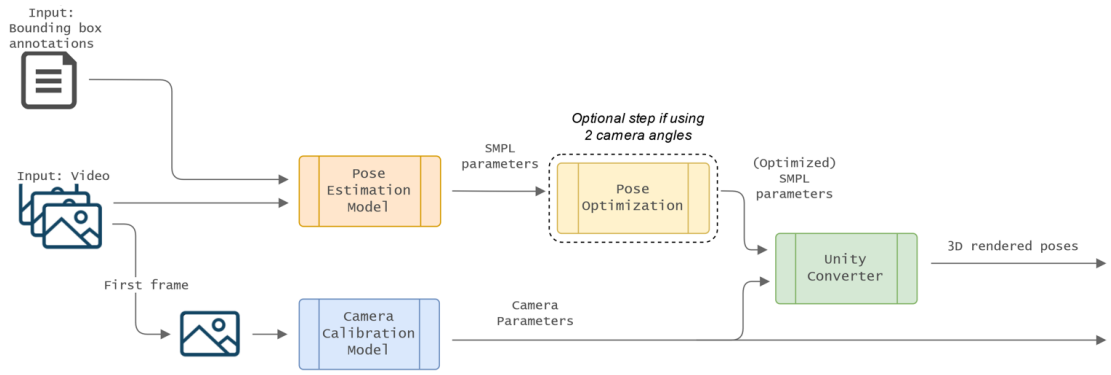


Figure 3.1: Visualization of the different modules in the pipeline.

Figure 3.2: Example frame of the input video recording of a football match.

3.2 Data

The project used annotated video recordings of football matches as data for the 3D pose estimation. All video recordings used a *static camera*, meaning the camera angle and position remained fixed throughout a video recording (but could vary between recordings). The resolution of the video recordings was 1920×1080 pixels and was taken from a high altitude such that the video covered a large part of the football field as visualized in Fig. 3.2.

Each frame of a video had accompanying bounding box annotations with one bounding box for each player visible in the frame. Fig. 3.3 visualizes the annotated bounding boxes for a frame of the video recording.

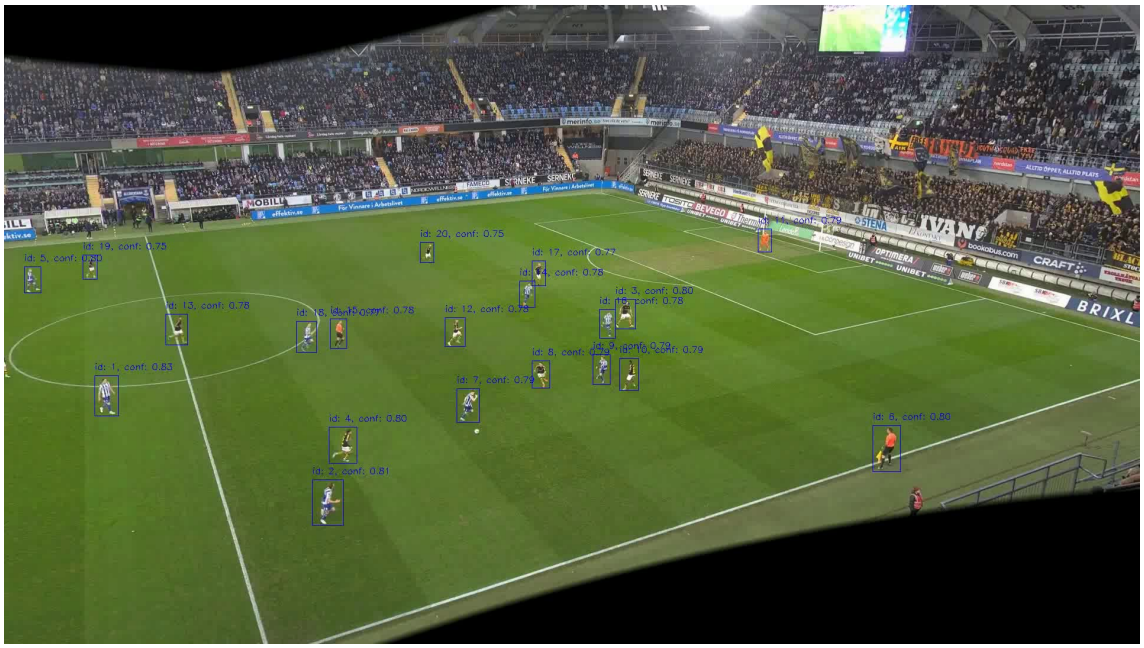


Figure 3.3: Example frame of the input video recording of a football match with accompanying annotated bounding boxes visualized.

Each bounding box also had an assigned ID to represent the identity of a player consistently across multiple frames. So, the annotated bounding box data is comprised of multiple entries formatted as:

`frame_id, object_id, center_x, center_y, width, height`.

Using the annotations and input video, image patches containing the visible players could be extracted. Notably, each bounding box could vary in size depending on the size of the player in the image. Each bounding box was initially padded to a $3:4$ aspect ratio. The part of the image contained in the bounding box was cut out as an image patch. Each extracted image patch of a player was resized to resolution 192×256 pixels using linear interpolation. Note that the bounding box was padded to the correct aspect ratio before the image patch was cut out and subsequently resized.

The resized resolution was the required size for the pose estimation model used in the subsequent stages. Fig. 3.4 shows the process of extracting and resizing bounding boxes. To visualize the results in this report, images of the football field (during the same match) from two different angles were used. These images are shown in Fig. 3.5.

Figure 3.4: (a) the visualized bounding box of a selected player; (b) the extracted image patch of the player using the bounding box annotation; (c) the resized image patch after linear interpolation.

(a) First camera view example.

(b) Second camera view example

Figure 3.5: The two camera view angles used as examples throughout the report.

3.3 Camera Calibration

Camera calibration was a crucial aspect of the project. It is required to be able to orient and place the players correctly on the football field. To be able to calibrate a camera, one needs to know both the intrinsic and extrinsic parameters of the camera. These are retrievable if the specifications of the camera used in the recording are known. If this information is unavailable, it generally has to be approximated instead. There are options for doing this if one has access to multiple cameras viewing the same object, or (as is true in this project) knows the specific dimensions of objects in the produced images.

Football fields used in international professional settings have standardized dimensions which are 100 to 110 meters long and 64 to 75 meters wide, as specified by the sport's governing bodies such as IFAB (The International Football Association Board) [34]. These field lines are visualized in Fig. 3.6 and ultimately enabled the possibility of automatic camera calibration estimation.

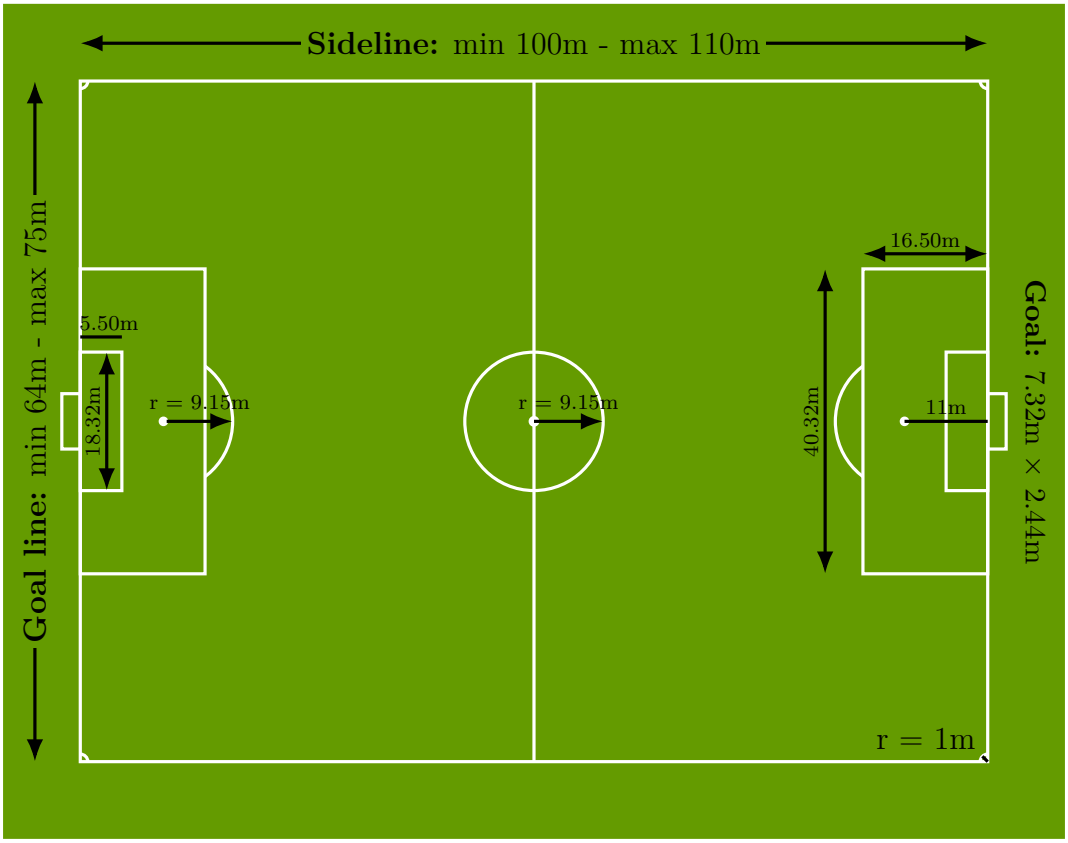


Figure 3.6: The dimensions of an international football field as described by the football field regulations "Laws of the game: Law 1" by IFAB.

Parameter	Description
`pan_degrees`	Scalar (degrees)
`tilt_degrees`	Scalar (degrees)
`roll_degrees`	Scalar (degrees)
`position_meters`	Vector (XYZ in meters)
`x_focal_length`	Scalar (pixels)
`y_focal_length`	Scalar (pixels)
`principal_point`	2D Point (XY coordinates in pixels)

Table 3.1: Camera parameters that are predicted by SoccerSegCal.

The SoccerSegCal model by Spiideo was used for camera calibration as it was specifically tailored for camera calibration in the context of football. The first frame was extracted from the video recording for which the camera parameters were to be derived. This single frame was used as the input for the calibration model and using this image the model estimates the different camera parameters listed in Table 3.1. The coordinate system assumed by the SoccerSegCal model is visualized in Fig. 3.7.

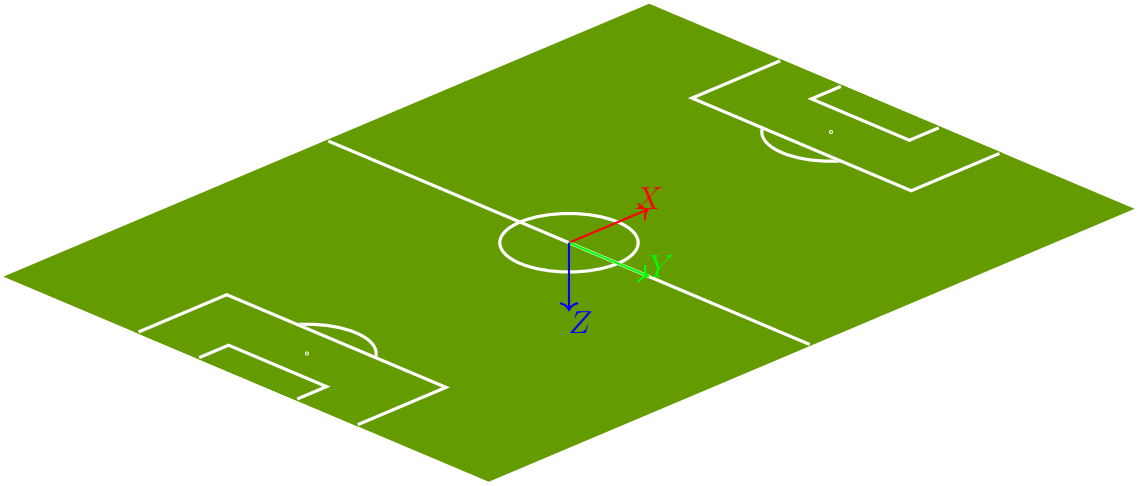


Figure 3.7: The coordinate system assumed by the SoccerSegCal model. Notably, the origin is located at the center of the field, the X-axis is parallel with the long side of the field, the Y-axis is parallel with the short side of the field, and the positive Z-axis points downwards orthogonal to the field.

Since the dimensions of the football field and the coordinate system are known, it was possible to validate the camera calibration visually. This could be done by creating a rough model of the field in 3D by placing points along the outer lines of the field, using the dimensions as seen in Fig. 3.6. The 4 corners of the 105×68 meter football field are denoted by $(\pm 52.5, \pm 34, 0)^\top$ and the midway line by $(0, \pm 34, 0)^\top$. The visual validation is performed by projecting these lines using the predicted camera parameters and seeing how well the lines align with the actual field lines in the image.

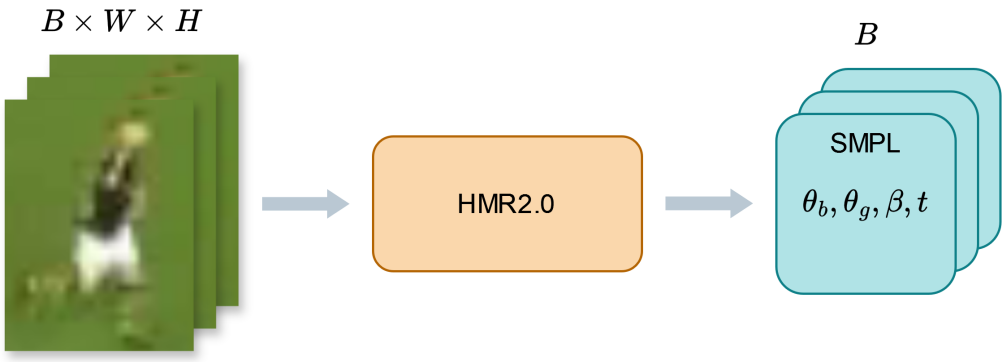


Figure 3.8: The HMR2.0 model processes a batch of images of size B, each with width W and height H, and produces a set of SMPL pose parameters and camera translations for each image in the batch.

3.4 Pose Estimation

The HMR2.0 model was used for pose estimation. The pre-trained model from the research paper was utilized and the input and output of the model are visualized in Fig. 3.8. It takes a batch of images containing human subjects, each with a resolution of 192×256 pixels, and produces a human pose prediction for each image. The input images were normalized and batched before being fed to the model. It should be noted that each frame was processed sequentially and separately; each frame contained roughly 20 players (depending on how many were visible in the frame) and the batch consisted of the extracted image patches of all players in the frame.

HMR2.0 uses the SMPL model for pose representation and the output is subsequently expressed as SMPL parameters along with additional camera parameters. The model predicts and outputs a set of SMPL parameters $(\boldsymbol{\beta}, \boldsymbol{\theta}_b, \boldsymbol{\theta}_g)$ and a camera translation $\boldsymbol{t}$ for each of the image patches in the input batch. As mentioned in subsection 2.2.2, the SMPL model allows extraction of 3D joints J given a predicted SMPL pose. The joints inferred by the SMPL pose are centered on the hip keypoint. The 2D points $(\hat{x}, \hat{y})^\top$ of the projected joints in the bounding box were obtained by projecting the 3D joints J:

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \pi(J). \tag{3.1}$$

The actual location of the projected points $(x, y)^\top$ in the full image were subsequently obtained by:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} w & h \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} + \begin{pmatrix} x_c \\ y_c \end{pmatrix}, \tag{3.2}$$

where w and h are the width and height of the input bounding box (before resizing) and $(x_c, y_c)^\top$ is the center point of the input bounding box in the image. So, from the pose estimation step, SMPL parameters representing the poses were extracted. Additionally, 3D joint coordinates and their 2D projections onto the image were obtained.

3.5 Player Localization

After the initial pose estimation, the next step involved 3D localization of every football player on the field, i.e., positioning the players relative to each other in the world coordinate system. This allowed the players to be placed in a 3D environment and subsequently capture their movement across the football field over time as seen in the original input video recording. Given that the relative positions of the joints in the pose were known, determining the 3D location of just one joint sufficed to place the entire model in the world. The hip keypoint was chosen to establish the SMPL model's position. Several approaches for the 3D localization of players were explored and will be described in this section.

3.5.1 Projecting 3D Location from Monocular Video

In the context of monocular camera settings, the task of positioning players and their estimated poses relies heavily on leveraging the intrinsic and extrinsic parameters derived from camera calibration, as explained in section 3.3. These parameters detail how the camera is rotated and positioned in relation to the center of the football field. Using this information, it was possible to reverse the projection of the hip keypoint on the image plane and ultimately retrieve the 3D placement of the player.

As mentioned in subsection 2.1.1, projecting a point involves converting a 3D point to its corresponding 2D point on the camera's image plane. Conversely, unprojecting involves deducing the possible 3D points that project to the specific 2D point in the image plane. This process inherently produces ambiguous results, resulting in a 3D line where all points project to the given 2D point.

To accurately determine the hip keypoint's 3D position in world coordinates, the critical assumption was made that all players were standing on the field, i.e., no players were in the air. Thus, the 3D placement of the hip keypoint was determined by unprojecting the 2D hip keypoint along the resulting ray, until it reached the specific hip height of the estimated pose.

Given the point of the hip keypoint on the image plane $(x, y)^\top$, the corresponding 3D ray $\boldsymbol{v}$ resulting from unprojection is calculated using the equation:

$$\boldsymbol{v} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = R^{-1}K^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \tag{3.3}$$

where K denotes the camera's intrinsic matrix, and R its rotation matrix.

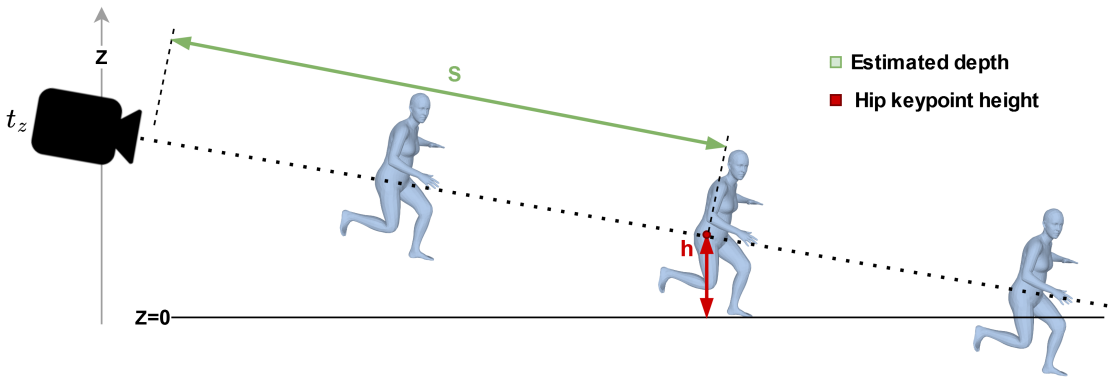


Figure 3.9: Positioning players on the football field at a target hip height.

This resulting ray $\boldsymbol{v}$ is the unnormalized direction aimed from the camera to the hip keypoint. This vector is then normalized to obtain the directional unit vector along which the hip keypoint is projected:

$$\boldsymbol{u} = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \frac{\boldsymbol{v}}{|\boldsymbol{v}|}. \tag{3.4}$$

The distance s along this vector to reach the actual 3D point of the hip is determined by:

$$s = \frac{h - t_z}{v_z}, \tag{3.5}$$

where t_z is the translation component along the z-axis of the camera translation $\boldsymbol{t}$, and h is the known hip height. Finally, the precise 3D location $\boldsymbol{p}$ of the hip keypoint is calculated as:

$$\boldsymbol{p} = \boldsymbol{t} + s \cdot \boldsymbol{u}, \tag{3.6}$$

thus establishing its position in world coordinates. The idea behind this projection of a keypoint along the camera ray is visualized in Fig. 3.9.

This approach determined the 3D locations of players within the world coordinate system, based on the assumption that players are in contact with the ground at all times. However, this assumption does not hold for airborne players, such as during jumps, where their positions are projected along the directional ray until they appear to be on the ground.

3.5.2 Player Ground Detection

To further improve the accuracy, the possibility of detecting when players touch the ground was explored. Determining whether a player is touching the ground or not from a single frame is difficult, so the temporal information across two frames was utilized. Movement was detected by measuring the distance of a player's joint in the image plane over two consecutive frames.

More specifically, the projected locations of the foot joints for a player were used to determine whether the feet were moving or not. Given the coordinates of a

foot keypoint in two consecutive frames, denoted as (x_1, y_1) for the first frame and (x_2, y_2) for the second frame, the motion of the foot is determined by calculating the Euclidean distance between these two points. The foot is considered to be moving if this distance exceeds a predefined pixel threshold δ. This is expressed as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}, \tag{3.7}$$

where the foot was deemed to be moving if $d > \delta$. Subsequently, a foot is assumed to not be moving if it is touching the ground. This was calculated for all pairs of consecutive frames the player appeared in.

As the foot of a player touches the ground, the hip of the player is subsequently projected along the directional ray until the player is placed on the field (as described in the previous section). In the cases where neither foot touched the ground, the player's position was determined by looking at the previous or next frame the player was touching the ground and linearly interpolating the player's position between them. This interpolation was achieved by listing the possible scenarios when a missing data entry was encountered for a player's hip location:

1. There exists a previous and a succeeding data entry.
2. There are no previous data entries but a succeeding value exists.
3. There exists a previous entry but no succeeding entries.
4. There does not exist any previous or succeeding entries.

In case number 1, the player has a previous and a succeeding known hip location. In this scenario, linear interpolation was applied to estimate the missing hip locations $\boldsymbol{p}(t)$ at a proportion t between the two known points $\boldsymbol{p}_1 = (x_1, y_1, z_1)^\top$ and $\boldsymbol{p}_2 = (x_2, y_2, z_2)^\top$ given by:

$$\boldsymbol{p}(t) = \boldsymbol{p}_1 + t \cdot (\boldsymbol{p}_2 - \boldsymbol{p}_1), \tag{3.8}$$

where $\boldsymbol{p}_1$ is the previous known point and $\boldsymbol{p}_2$ is the next known point. Moreover, $0 \leq t \leq 1$ is the interpolation parameter corresponding to the relative position between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, and calculated as:

$$t = \frac{k - i}{j - i}, \tag{3.9}$$

where k is the current index, i is the index of the first missing value, and j is the index of the next known value.

In case number 2, all the missing hip locations between the current missing entry and the known succeeding entry were assigned the value of the succeeding frame. Similarly, the reverse approach was applied for case number 3, where the known previous value was assigned to all subsequent unknown frames. The fourth case, although uncommon, assumed a default hip height of 0.85 meters to estimate the 3D hip location from the 2D point in the camera's image plane. This methodology is detailed in algorithm 1.

Algorithm 1: Interpolation of 3D Hip Location

```
Data: Player Data
Result: Interpolated 3D Hip Locations

foreach player in Player Data do
    Initialize previous location as None;
    foreach frame in player data do
        if hip location in current frame exists then
            Update previous location to current frame location;
            Continue to next frame;
        end
        else
            Find the index of the next available hip location;
            if Previous location and succeeding location exist then
                Interpolate linearly between start and end data points;
            end
            else if No previous location but a succeeding location exists then
                Fill missing hip locations with the succeeding frame's value;
            end
            else if Previous location exists but no succeeding location then
                Fill the missing hip locations with the previous value;
            end
            else
                Set default hip height to 0.85 meters;
                Estimate the 3D position from the known 2D image-point;
            end
        end
    end
end
```

3.5.3 Triangulation from Multi-View Footage

An approach for 3D localization utilizing perspectives from two different cameras was also explored. Given that both cameras were calibrated, triangulation was possible for determining the 3D coordinates of the hip joint. In addition, the assumption that the cameras were not filming in parallel was made to allow for triangulation. This assumption holds true due to the placement of the cameras used to record IFK's videos.

The triangulation involved unprojecting the hip keypoint from each camera's frame to retrieve two lines that correspond to the projected positions of the hip keypoints. However, due to potential inaccuracies in both camera calibration and the projection of the 2D hip keypoints, these lines might not intersect exactly. Therefore, the triangulation was refined by identifying the closest points on each line and computing the midpoint between these two points as the triangulated position.

Let $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ be the directional vectors (calculated similarly to Equation 3.3) from the first and second camera originating from the camera centers $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$.

The lines can be expressed as:

$$\text{Line 1:} \quad \boldsymbol{l}_1 = \boldsymbol{c}_1 + s\boldsymbol{d}_1, \tag{3.10}$$

$$\text{Line 2:} \quad \boldsymbol{l}_2 = \boldsymbol{c}_2 + t\boldsymbol{d}_2. \tag{3.11}$$

The closest points on each line to the other line can be found by solving the optimization problem:

$$\min_{s,t} \|(\boldsymbol{c}_1 + s\boldsymbol{d}_1) - (\boldsymbol{c}_2 + t\boldsymbol{d}_2)\|^2, \tag{3.12}$$

where s and t scale the respective vectors. Note that $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are not parallel due to the different viewing angles of the two cameras. The solution to this problem identifies the scalar parameters s and t that minimize the squared Euclidean distance between points on the two lines. Note that the distance is squared to simplify the calculations as it yields the same solutions.

So the objective is to minimize the squared Euclidean distance between points on these lines:

$$D^2 = \|\boldsymbol{l}_1 - \boldsymbol{l}_2\|^2 = \|\boldsymbol{c}_1 + s\boldsymbol{d}_1 - \boldsymbol{c}_2 - t\boldsymbol{d}_2\|^2. \tag{3.13}$$

Expanding and simplifying D^2 gives:

$$D^2 = (\boldsymbol{c}_1 - \boldsymbol{c}_2 + s\boldsymbol{d}_1 - t\boldsymbol{d}_2) \cdot (\boldsymbol{c}_1 - \boldsymbol{c}_2 + s\boldsymbol{d}_1 - t\boldsymbol{d}_2). \tag{3.14}$$

To find the minimum, set the partial derivatives with respect to s and t to zero:

$$\frac{\partial D^2}{\partial s} = 2\left(\boldsymbol{d}_1 \cdot (\boldsymbol{c}_1 - \boldsymbol{c}_2 + s\boldsymbol{d}_1 - t\boldsymbol{d}_2)\right) = 0, \tag{3.15}$$

$$\frac{\partial D^2}{\partial t} = -2\left(\boldsymbol{d}_2 \cdot (\boldsymbol{c}_1 - \boldsymbol{c}_2 + s\boldsymbol{d}_1 - t\boldsymbol{d}_2)\right) = 0. \tag{3.16}$$

These equations can be arranged into the matrix form:

$$\begin{pmatrix} \boldsymbol{d}_1 \cdot \boldsymbol{d}_1 & -\boldsymbol{d}_1 \cdot \boldsymbol{d}_2 \\ \boldsymbol{d}_1 \cdot \boldsymbol{d}_2 & -\boldsymbol{d}_2 \cdot \boldsymbol{d}_2 \end{pmatrix} \begin{pmatrix} s \\ t \end{pmatrix} = \begin{pmatrix} (\boldsymbol{c}_2 - \boldsymbol{c}_1) \cdot \boldsymbol{d}_1 \\ (\boldsymbol{c}_2 - \boldsymbol{c}_1) \cdot \boldsymbol{d}_2 \end{pmatrix}, \tag{3.17}$$

where the matrix on the left includes dot products of direction vectors, which correspond to the coefficients of s and t in the derivative equations. Solving this system gives the scalar parameters s and t that locate the closest points on each line.

These values are then used to compute the closest points on each line:

$$\boldsymbol{p}_1 = \boldsymbol{c}_1 + s\boldsymbol{d}_1, \quad \boldsymbol{p}_2 = \boldsymbol{c}_2 + t\boldsymbol{d}_2. \tag{3.18}$$

The midpoint between these closest points is then calculated as:

$$\boldsymbol{p}_{triangulated} = \frac{\boldsymbol{p}_1 + \boldsymbol{p}_2}{2}, \tag{3.19}$$

providing a refined estimation of the 3D position of the hip.

3.6 Pose and Location Optimization

An optimization approach was explored to refine the estimated pose and 3D location of a player using footage from two different cameras. This method assumes that the predicted pose from each frame projects accurately onto the player from the viewpoint of the same camera. However, projective ambiguities can make these estimated poses appear inaccurate when viewed from the other camera's angle.

To address this, the optimization process treats the projections of the joints from the pose estimated by the same camera as a pseudo-ground truth. The player's pose is then refined by minimizing the L1 loss between the projections of the pose and the pseudo-ground truth considering both camera angles, thereby reducing reprojection errors across the joints. Additionally, the discriminator originally trained and used during the development of the HMR2.0 model was used as an adversarial loss. This was done to ensure the optimized parameters still represented a feasible human pose.

The optimization procedure worked as follows: first, the 3D pose of the player was predicted from each camera angle yielding two sets of SMPL parameters $(\boldsymbol{\beta}_1, \boldsymbol{\theta}_{b1}, \boldsymbol{\theta}_{g1})$ and $(\boldsymbol{\beta}_2, \boldsymbol{\theta}_{b2}, \boldsymbol{\theta}_{g2})$. The joints of the SMPL pose were also projected onto the player in their respective image, thus obtaining the aforementioned pseudo-ground truths $\boldsymbol{p}_1^*$ and $\boldsymbol{p}_2^*$. Then, the 3D location of the player's hip was triangulated as described in subsection 3.5.3, resulting in a location vector $\boldsymbol{h} \in \mathbb{R}^3$. The optimization was then performed using the estimated SMPL parameters from the second view (selected arbitrarily out of the two), i.e., $(\boldsymbol{\beta}_2, \boldsymbol{\theta}_{b2}, \boldsymbol{\theta}_{g1})$, the orientation for the first view $\boldsymbol{\theta}_{g1}$, and the triangulated world coordinates for the hip placement $\boldsymbol{h}$.

The minimization was expressed as:

$$\min_{\boldsymbol{\beta}_2, \boldsymbol{\theta}_{b2}, \boldsymbol{\theta}_{g2}, \boldsymbol{\theta}_{g1}, \boldsymbol{h}} \|\boldsymbol{p}_1^* - \pi_1(\delta(\boldsymbol{\beta}_2, \boldsymbol{\theta}_{b2}, \boldsymbol{\theta}_{g2}) + \boldsymbol{h})\|_1 + \|\boldsymbol{p}_2^* - \pi_2(\delta(\boldsymbol{\beta}_2, \boldsymbol{\theta}_{b2}, \boldsymbol{\theta}_{g1}) + \boldsymbol{h})\|_1 + \mathcal{D}(\boldsymbol{\beta}_2, \boldsymbol{\theta}_{b2}), \tag{3.20}$$

where π_1 and π_2 are the perspective projections for the two cameras, δ converts the SMPL parameters to corresponding predicted 3D joints, and $\mathcal{D}$ is the adversarial loss function. The `PyTorch` library [35] for `Python` was used for the technical implementation, and the Adam optimizer [36] was used as the optimization algorithm.

3.7 Animating 3D Poses

Following pose estimation, localization, and optimization the next step was to process this data in order to animate and render player meshes. Blender's Python API (BPY) was used to animate and render SMPL parameters along with 3D locations. This section will explain how this process was performed.

In order to take the SMPL data and bounding box data and create a single animated 3D file, several different processes were carried out. The first process was to import a default SMPL model as a `.fbx` file provided by Loper et. al. [17]. This imported model was then loaded into the scene using BPY.

The second process was carried out by positioning the model's bones according to the generated pose data from HMR2.0. This was done for every player and for each respective frame that the player had corresponding pose data generated. Rotating the bones was done by using normalized quaternion (i.e., unit quaternion) rotations. The pose data from $\boldsymbol{\theta}_b, \boldsymbol{\theta}_g$ was used to rotate each bone with its corresponding axis angle.

After the rotation was calculated for each bone it was also inserted into the current frame by a `keyframe insert` to mark that point in the animation timeline to define the armature's pose. The $\boldsymbol{\beta}$ parameters were iterated over to set the shape of the default model and then inserted similarly to the bone rotations. These two steps arranged the bones so the player was posed correctly according to the generated SMPL data.

The next part was to position the pose to its respective 3D world location, which was done by setting the location of the armature pelvis to the corresponding 3D hip location. The last step was correcting the axes inside Blender to match the final coordinate system for the reconstructed football field inside VR. This was done by applying three different quaternion rotations to every armature root and pelvis bone respectively.

After positioning and posing every player all their 3D data was exported to an appropriate file format. This was chosen as a `.gltf`-file since it has fast loading, low runtime processing required, and is supported by several game engines (including Unity). The pseudocode of the BPY script for loading, converting, and exporting all 3D data is listed in algorithm 2.

Algorithm 2: SMPL data animation and conversion to a 3D file format

Data: player_ids × Frames × $(\theta, \beta, hip_{3D})$ and camera calibration
Result: Rendered and animated poses exported as `.gltf`-file

`players` ← player_ids;
`player_data` ← $(Frames, (\boldsymbol{\theta}, \boldsymbol{\beta}))$;
`pelvis_3D` ← hip_{3D};
`x_90_QR` ← `Quaternion`$(1, 0, 0, 90°)$;
`x_180_QR` ← `Quaternion`$(1, 0, 0, 180°)$;
`y_pan_QR` ← `Quaternion`$(0, 1, 0,$ `pan_degree`$°)$;

foreach *player and data* $(p_i, p_{data}) \in$ (`players`, `player_data`) **do**
  Import fbx scene and neutral SMPL model into bpy;
  Create bpy obj and armature;
  foreach *Frame, hip location and SMPL data* $f_i, hip3D_i, \beta_i, \theta_i \in (p_i, p_{data},$ `pelvis_3D`$)$ **do**
    Location of *armature_pelvis* ← $(0, 0, 0)$;
    Quaternion rotation of *armature_root* ← `x_90_QR`;
    Insert keyframe with ← $(armature\_root, f_i)$;
    foreach *Bone and rotation* $(b_i, m_i) \in \theta_i$ **do**
      Quaternion rotation of $b_i \leftarrow m_i$;
      if b_i == *Pelvis bone* **then**
        Quaternion rotation of b_i ← `x_180_QR` × `y_pan_QR`;
      end
      Insert keyframe with ← (b_i, f_i); `//Inserting Rotated bones`
    end
    Insert Shape keyframe with ← (β_i, f_i); `//Inserting shape value`
    Location of *armature_pelvis* ← $hip3D_i$;
    Insert keyframe with ← $(armature\_pelvis, f_i)$;
  end
end
Export all bpy data to `.gltf`; `//Meshes and animations to .gltf`

4

Results

This chapter presents the visual results for camera calibration, pose estimation, player localization, optimization, and animated 3D poses. Due to the lack of annotations in the provided data by IFK, the results are mostly qualitative. Visual assessments are used to provide insights into the tasks' accuracies.

Note that when players are mentioned by ID, it refers to the annotated ID, unrelated to the players' jersey number. Results from poses and optimizations are demonstrated on a variety of different IDs while the results from player localization approaches and animations are visualized on IDs 4, 6, and 11. These IDs were chosen because they demonstrate different difficulties in their respective results and especially showcase difficulties in the monocular methods used. In addition to this, there were no ID switches between these players from Project 1's [4] bounding box annotations.

4.1 Camera Calibration

In Table 4.1 the predicted camera parameters for each of the example images mentioned in section 3.3 are listed. Fig. 4.1 shows the 3D points outlining the football field as described in section 3.3. These points are the ones that are projected onto the images afterwards using the calibrated cameras. The figure also visualizes the field's location relative to the coordinate system used by the camera calibration model.

Parameter	Value
pan_degrees	20.87
tilt_degrees	69.47
roll_degrees	0.33
position_meters	[-1.25, 55.56, -16.87]
x_focal_length	1101.50
y_focal_length	1101.50
principal_point	[480.00, 270.00]

Parameter	Value
pan_degrees	-98.94
tilt_degrees	58.11
roll_degrees	0.15
position_meters	[64.72, -2.58, -21.43]
x_focal_length	752.81
y_focal_length	752.81
principal_point	[480.00, 270.00]

Table 4.1: Predicted camera parameters from two cameras of the same field. The left table corresponds to the camera located at the long side of the football field, i.e., the first view. The right table corresponds to the camera located at the short side behind the goal, i.e., the second view.

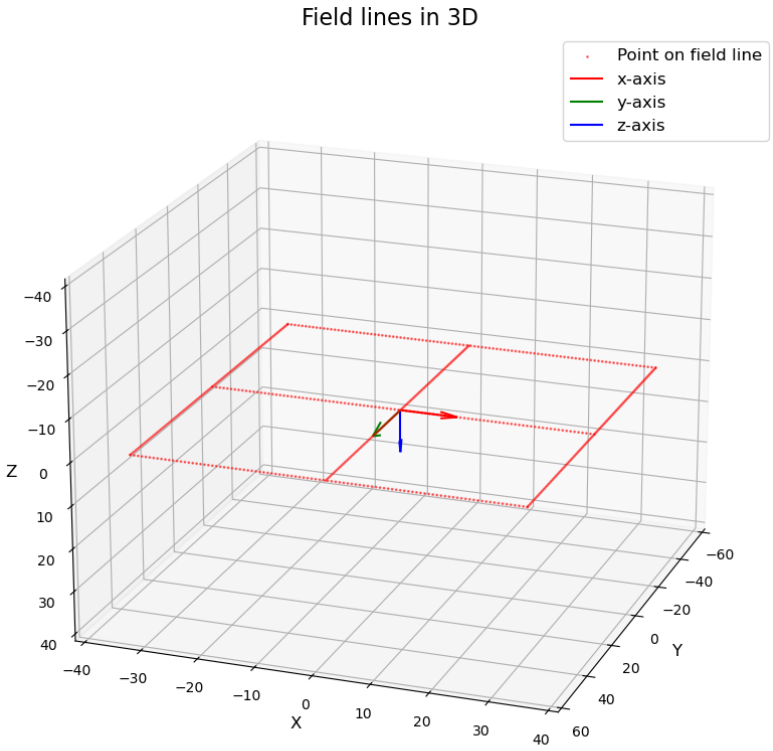


Figure 4.1: 3D plot showing the field lines of the football field and the coordinate system used by the camera calibration model.

Fig. 4.2 and Fig. 4.3 show where the 3D field lines project onto each of the images using the predicted camera parameters. The figures show that the projected points align relatively well with the intended field lines with slight inaccuracies. In the lower-left corner of Fig. 4.2 the inaccuracies are visible as the projected red dots do not align exactly with the white lines. In the ideal scenario, the projected red dots would be placed exactly on the white lines.

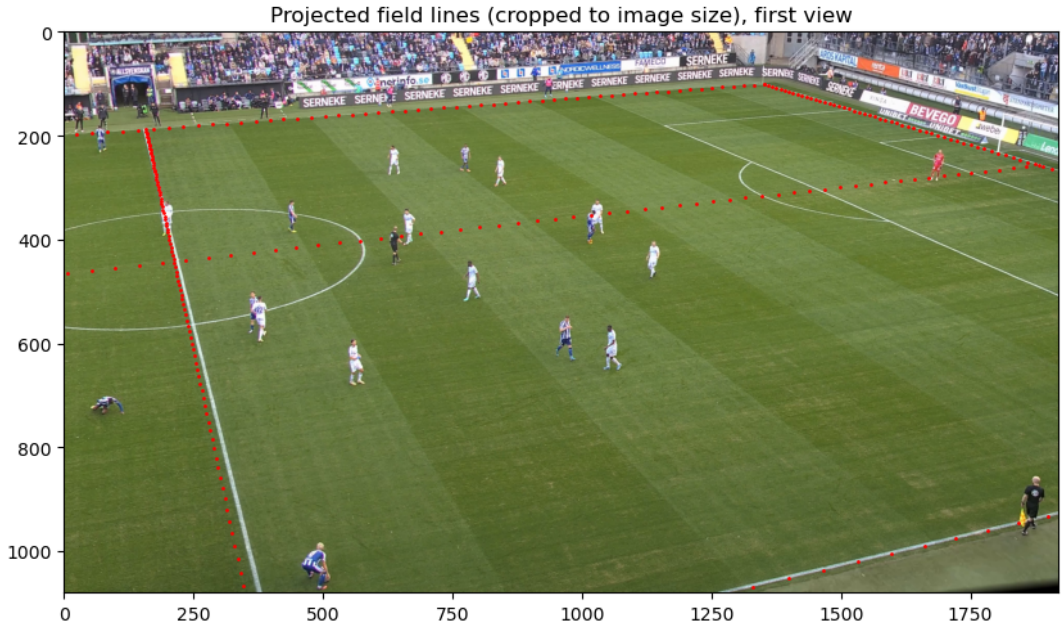


Figure 4.2: The red dots mark the projected field lines onto the image from the first camera view.

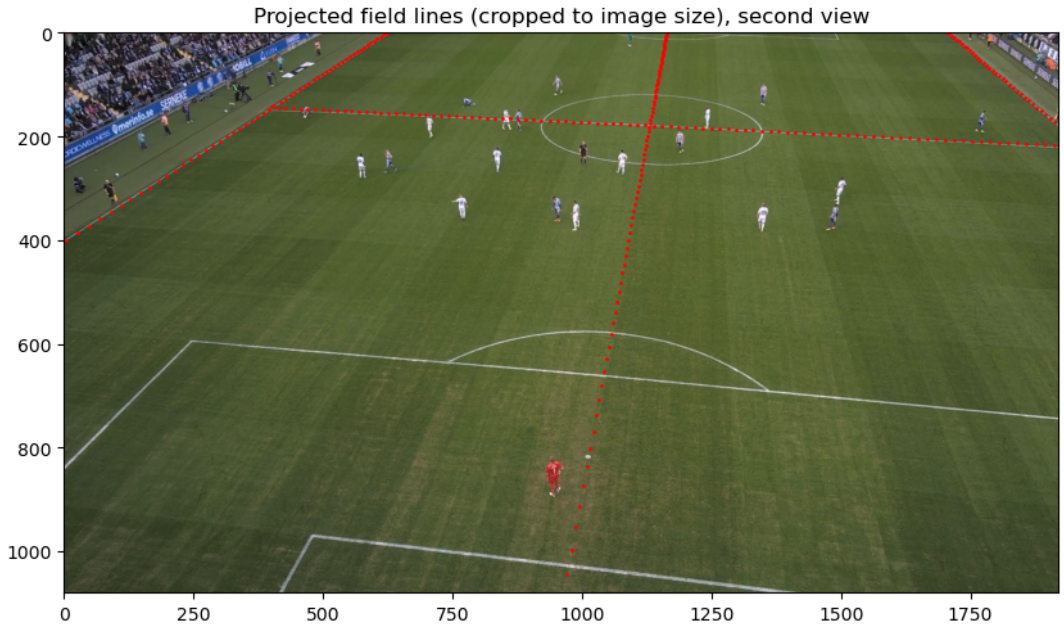


Figure 4.3: The red dots mark the projected field lines onto the image from the second camera view

Moreover, Fig. 4.4 and Fig. 4.5 show how the entire field would project onto the images. As can be seen, the field is notably distorted outside the image and grows in size. This means that any point not visible in the image would be projected with significant distortion. However, as only players and points visible in the images are used during this project this is not an immediate issue.

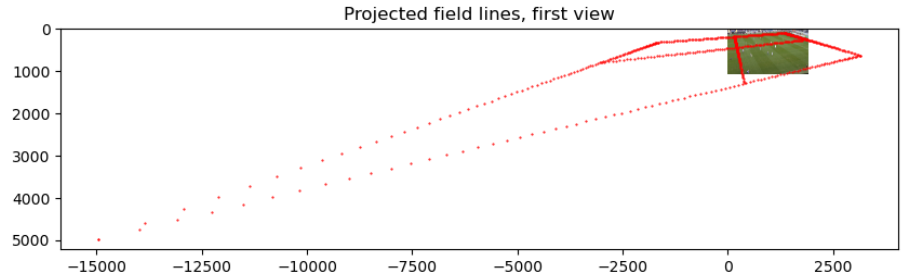


Figure 4.4: All of the projected field lines from the first camera view.

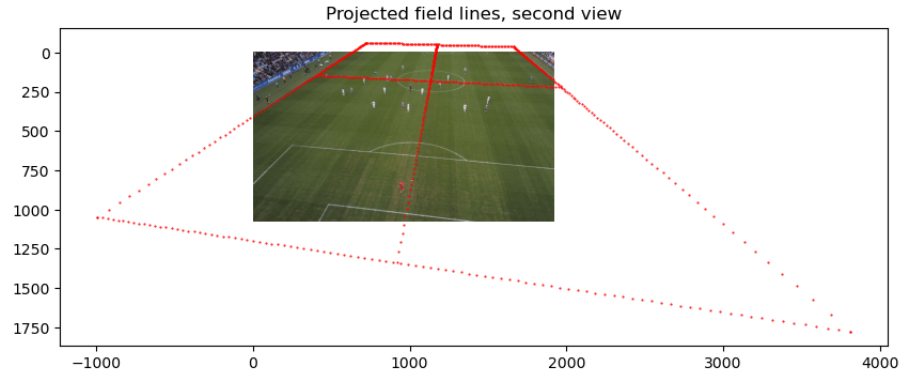


Figure 4.5: All of the projected field lines from the second camera view.

4.2 Pose Estimation

Fig. 4.6 and Fig. 4.7 show a sample of pose estimations by HMR2.0. For each column in the figure, the rows show the input bounding box (after resizing), a projection of the predicted pose by HMR2.0, the projected SMPL joints for the predicted pose, and finally the same predicted pose from the side. Fig. 4.6 shows the majority of cases where players are isolated in upright positions whereas Fig. 4.7 is meant to highlight special cases such as unusual poses, occlusion, or contact between players.

Figure 4.6: A selection of different poses and their subsequent pose estimates. Each column is a pose estimate case. The first row shows the original input bounding box after resizing. The second row shows the SMPL pose estimated by HMR2.0 projected onto the bounding box. The third row shows the projected SMPL joints. The final row shows a side view of the SMPL pose.

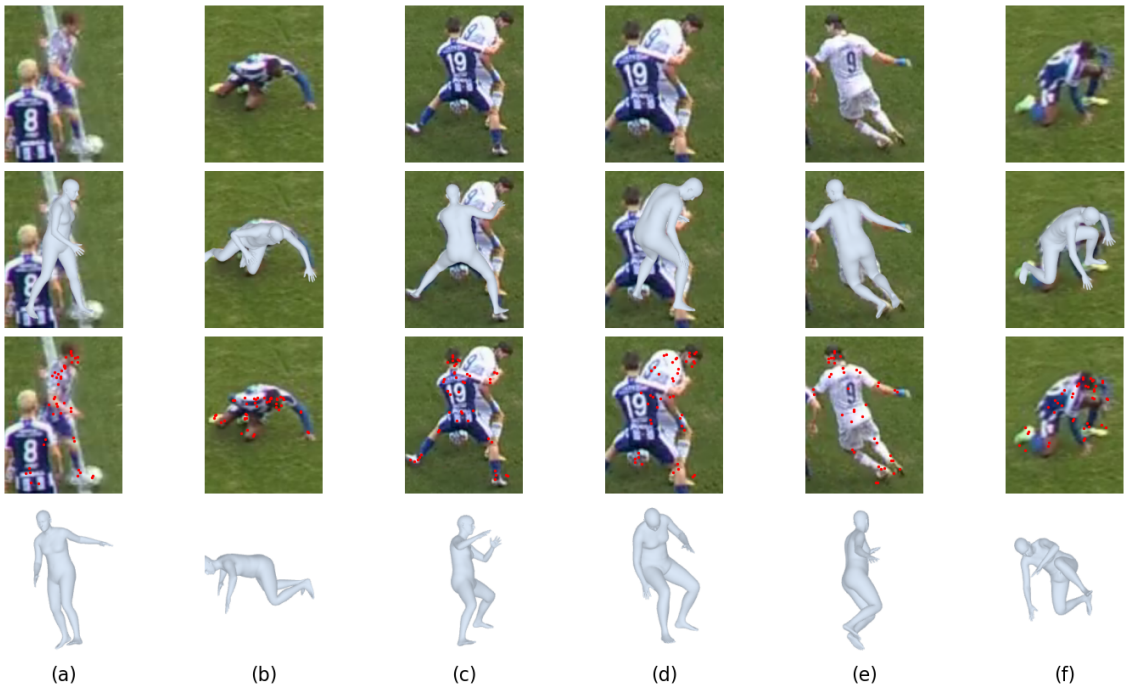


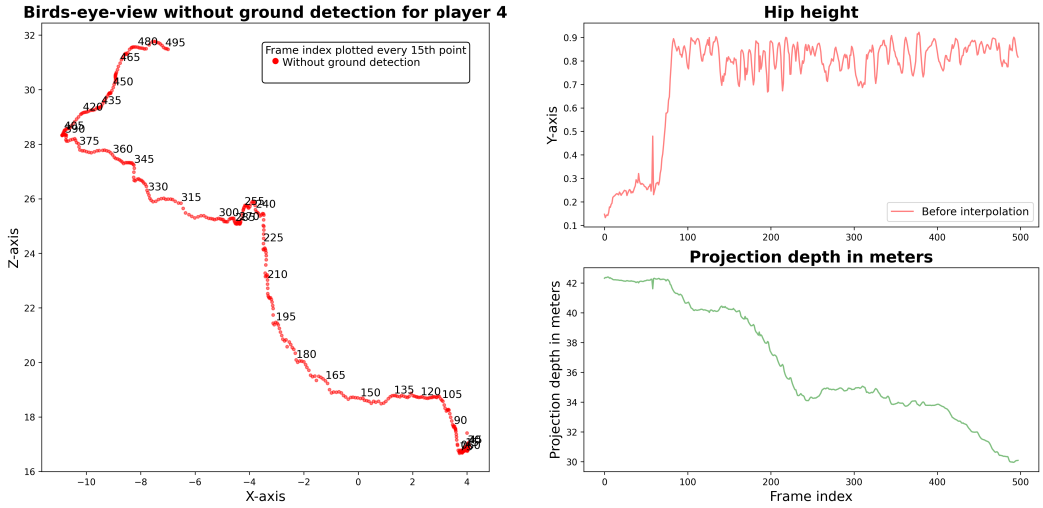
Figure 4.7: Another selection of different poses and their subsequent pose estimates. This figure adheres to the same format as the previous SMPL pose visualization.

4.3 Player Localization

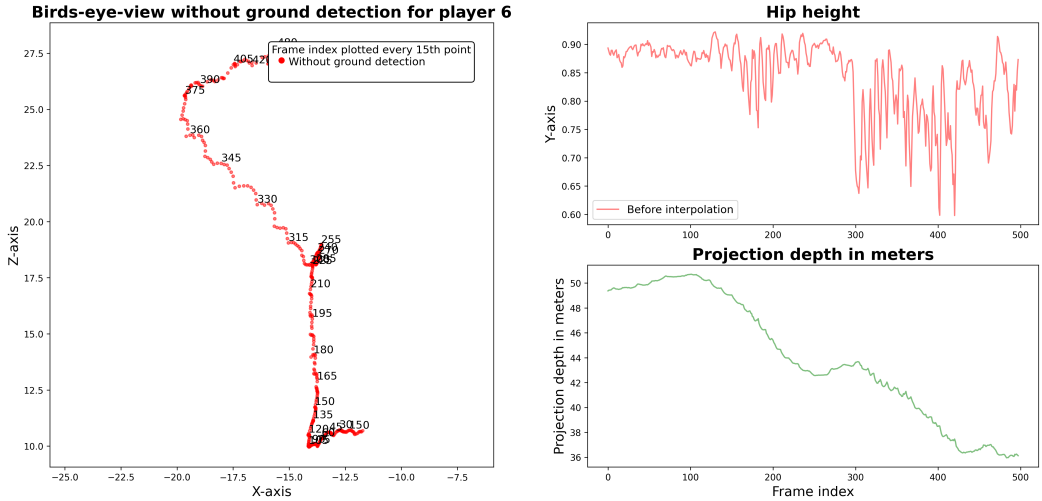
This section presents the three main approaches used to position the players in 3D. It covers the results of 3D positioning by projecting the players from the camera's view to the ground. Additionally, it includes results from detecting when the players' feet touch the ground, positioning them at those moments, and then interpolating their positions in frames where their feet do not touch the ground. Finally, the results of 3D positioning by triangulating the hip keypoint are also presented.

4.3.1 Projecting 3D Location from Monocular Video

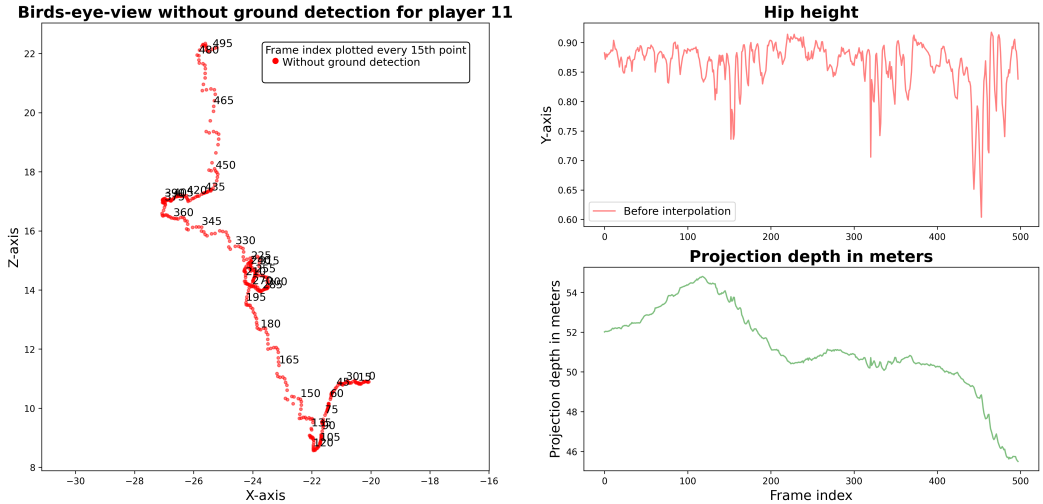
Results from projecting players' hips are presented in Fig. 4.8. The plots to the left visualize players' 2D movements from a birds-eye-view perspective where the Z-axis and X-axis are axes inside Blender's coordinate system and the unit is in meters. Additional frame numbers are visualized on every 15:th 2D point plotted, illustrating the direction of 2D movement. On the right side, there are two plots for every player visualizing the hip height and depth from the hip to the camera, also denoted in meters. The depths are visualized as green graphs and positioned below each plot of hip height.



(a) Player 4 monocular 3D placement



(b) Player 6 monocular 3D placement



(c) Player 11 monocular 3D placement

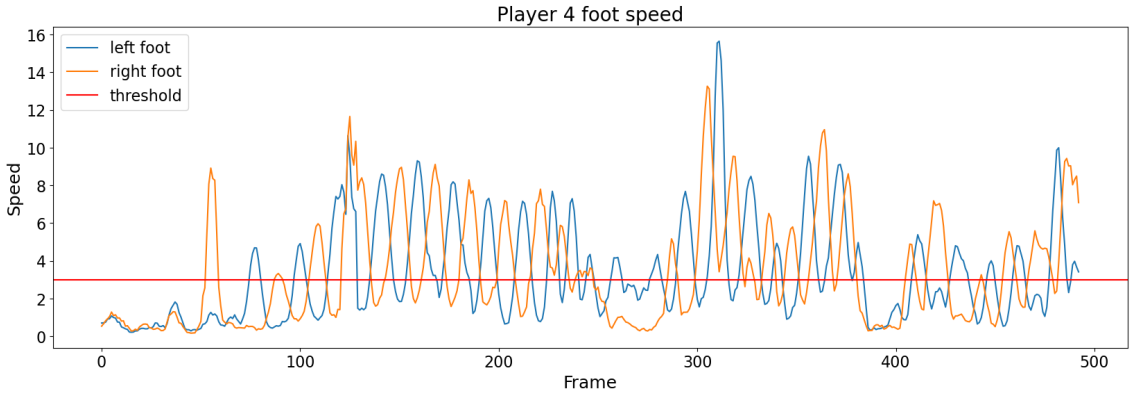
Figure 4.8: Plots visualizing three distinct players' 3D locations. 2D positions are presented to the left side from a birds-eye-view perspective. The right side shows each player's noise in hip heights and depth estimations.

4.3.2 Player Ground Detection

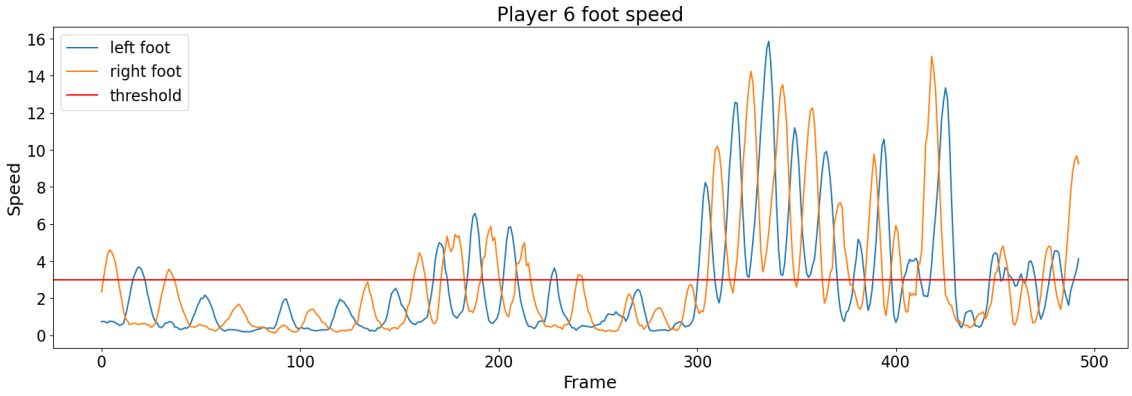
In Fig. 4.9, the speeds of three different players' feet throughout the video are visualized. The speeds represent the pixel difference in the projected foot joint positions between two consecutive frames. The graphs also display a movement threshold: feet with speeds below this threshold are considered to be touching the ground, while feet with speeds above it are considered to be moving in those specific frames.

To further visualize whether the feet are touching the ground or moving, the bounding boxes for each player have been extracted for 10 specific frames. This is visualized in Fig. 4.10. To indicate foot movement status, a red or green dot has been placed on the player's foot joint: red signifies that the foot is touching the ground, while green signifies that the foot is moving.

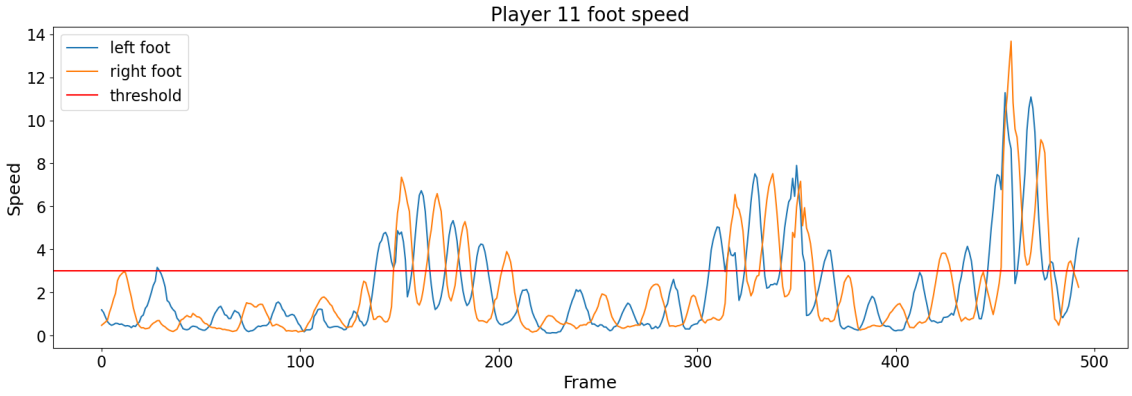
The resulting 3D positions for the players are illustrated in Fig. 4.11. To the left, the 2D birds-eye-view perspective is plotted with corresponding frame indices every 15:th frame with the red dots as the detected locations where the movement speed was below the threshold of 3. The blue dots are positions from linear interpolation. Hip height and the camera-to-player's hip depth are shown on the right side.



(a) Player 4's feet.



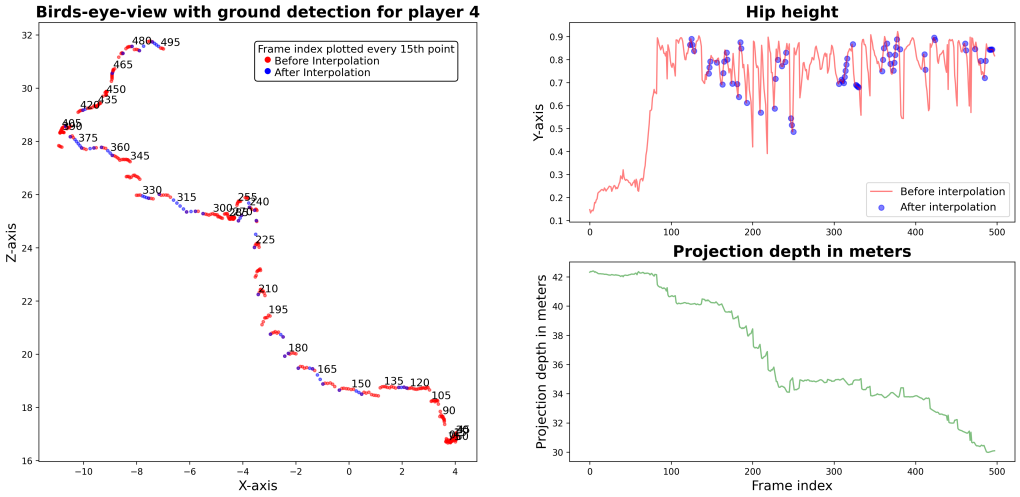
(b) Player 6's feet.



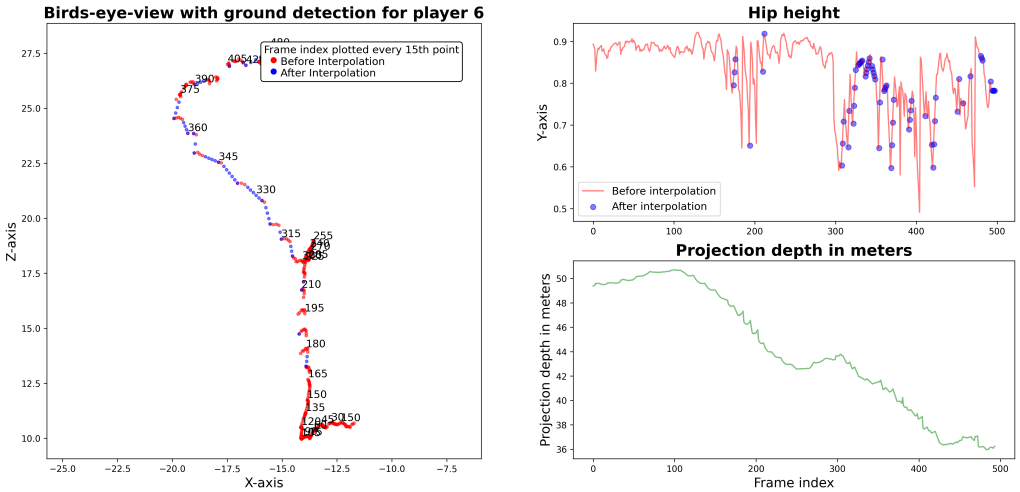
(c) Player 11's feet.

Figure 4.9: The speeds of players' feet, measured in pixels, for each frame of the video. The movement threshold is also shown: feet with speeds above this threshold are considered to be moving, while feet with speeds below it are considered to be touching the ground.

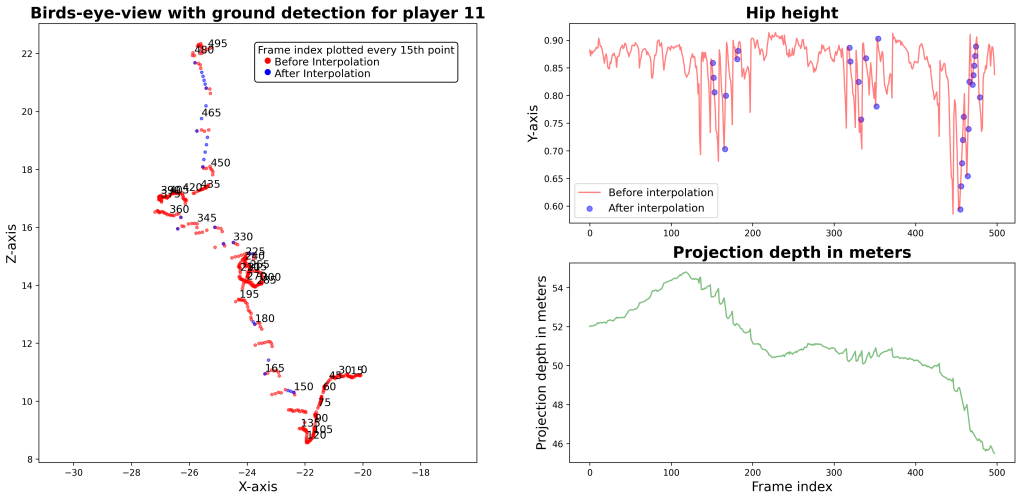
Figure 4.10: Visualization of players' foot movement status across 10 specific frames. Each subfigure shows the bounding box of a player, with red dots indicating that the foot is touching the ground and green dots indicating that the foot is moving, based on the previously discussed movement threshold.



(a) Player 4 ground detection 3D placement



(b) Player 6 ground detection 3D placement

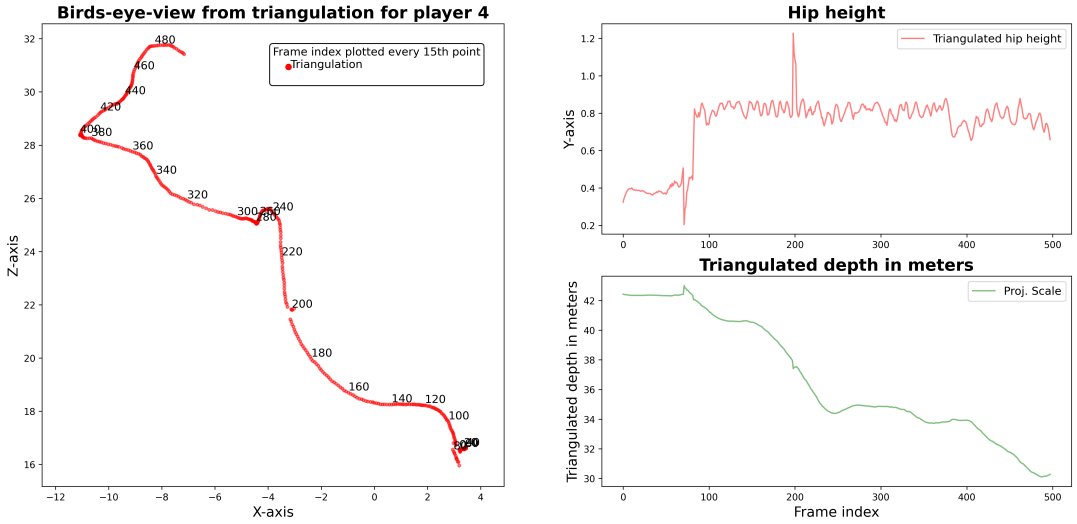


(c) Player 11 ground detection 3D placement

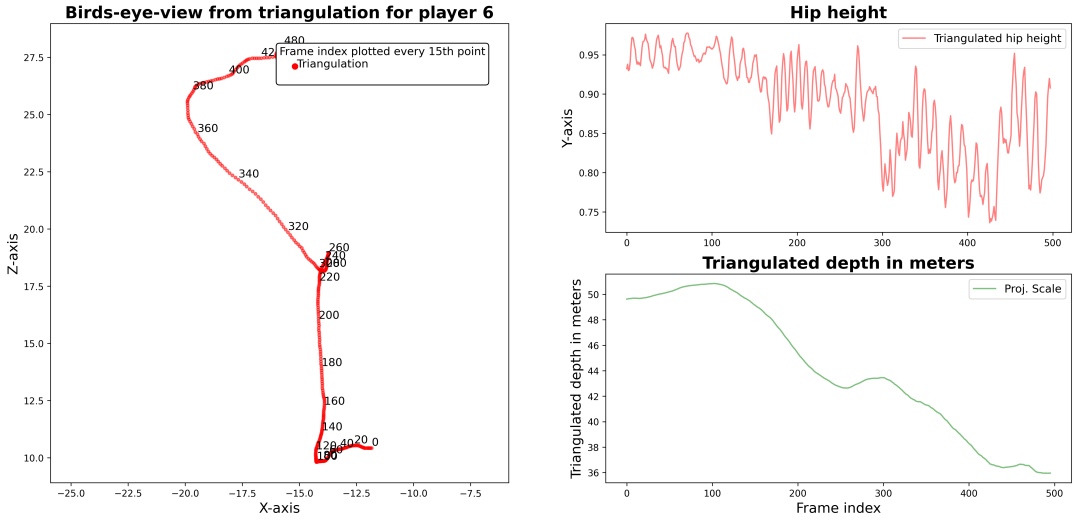
Figure 4.11: Plots visualizing three distinct players' 3D locations derived from ground detection and interpolation. Created with movement threshold of 3.

4.3.3 Triangulation from Multi-View Footage

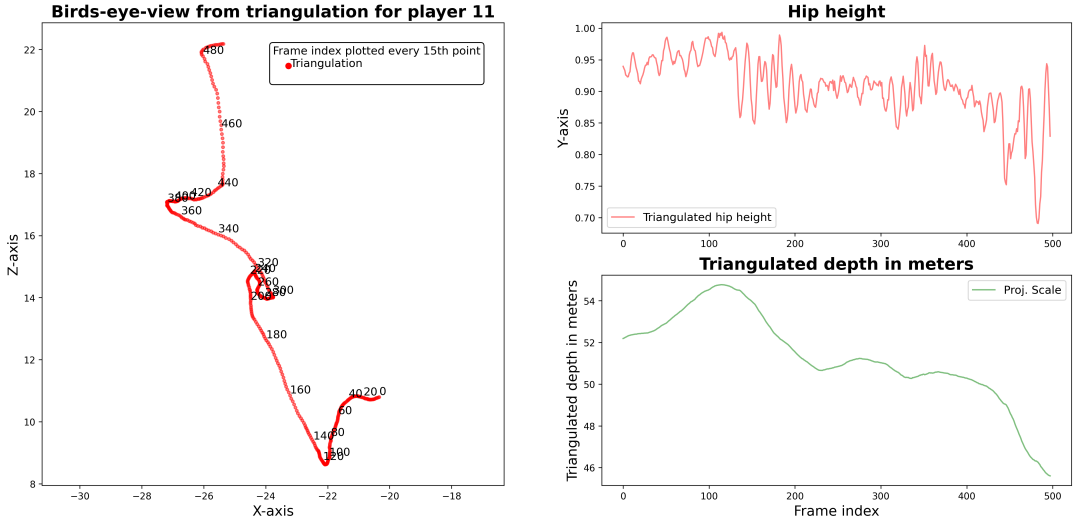
Results for triangulated 3D positions can be seen in Fig. 4.12. All the presented graphs have been derived from triangulation. Triangulation was performed for every frame, i.e., no ground detection nor interpolation was performed.



(a) Player 4 triangulated 3D location



(b) Player 6 triangulated 3D location



(c) Player 11 triangulated 3D location

Figure 4.12: Triangulated 3D locations from two different views. ID associations were done manually across the two views for every player visualized.

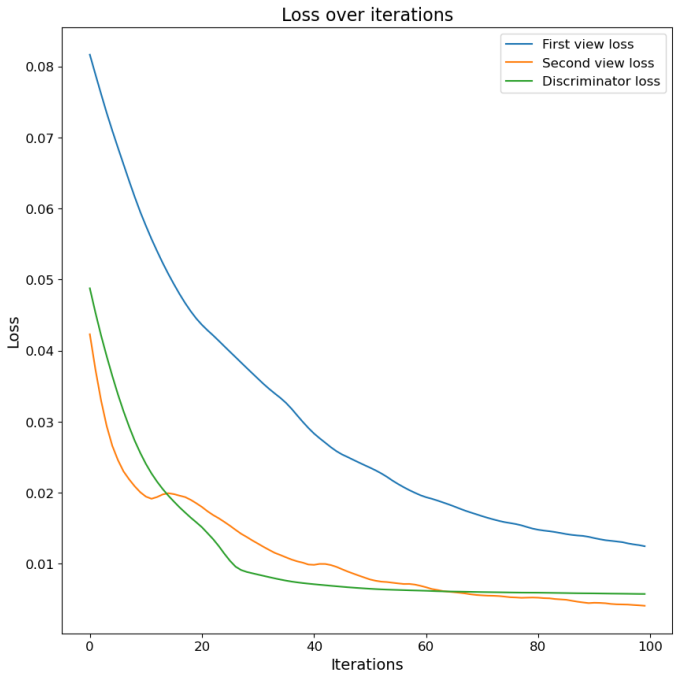


Figure 4.13: The graph shows the First view loss (blue), Second view loss (orange), and Discriminator loss (green) over 100 iterations, showing a steady decrease and convergence of all three losses.

4.4 Pose and Location Optimization

Since the pose and position optimization requires ID association of players between views, the resulting pose- and position optimization of a single player is presented.

The resulting loss during the optimization is visualized in Fig. 4.13. It shows the three different losses used: L1 loss for the projections in the first and second view, as well as the adversarial loss from the discriminator. The loss used during the training is the cumulative loss of these three. This plot illustrates the training process, showing how the different losses decrease and stabilize over time, indicating the progress and convergence of the optimization.

Moreover, Fig. 4.14 shows the projected joints of the SMPL pose before and after optimization for the two different views. It also shows the pseudo-truth joints used for the optimization. These pseudo-truth joints are marked as the blue dots while the projected joints of the predicted SMPL pose being optimized correspond to the red dots. The numbers next to each of the projected points are the IDs of the different joints.

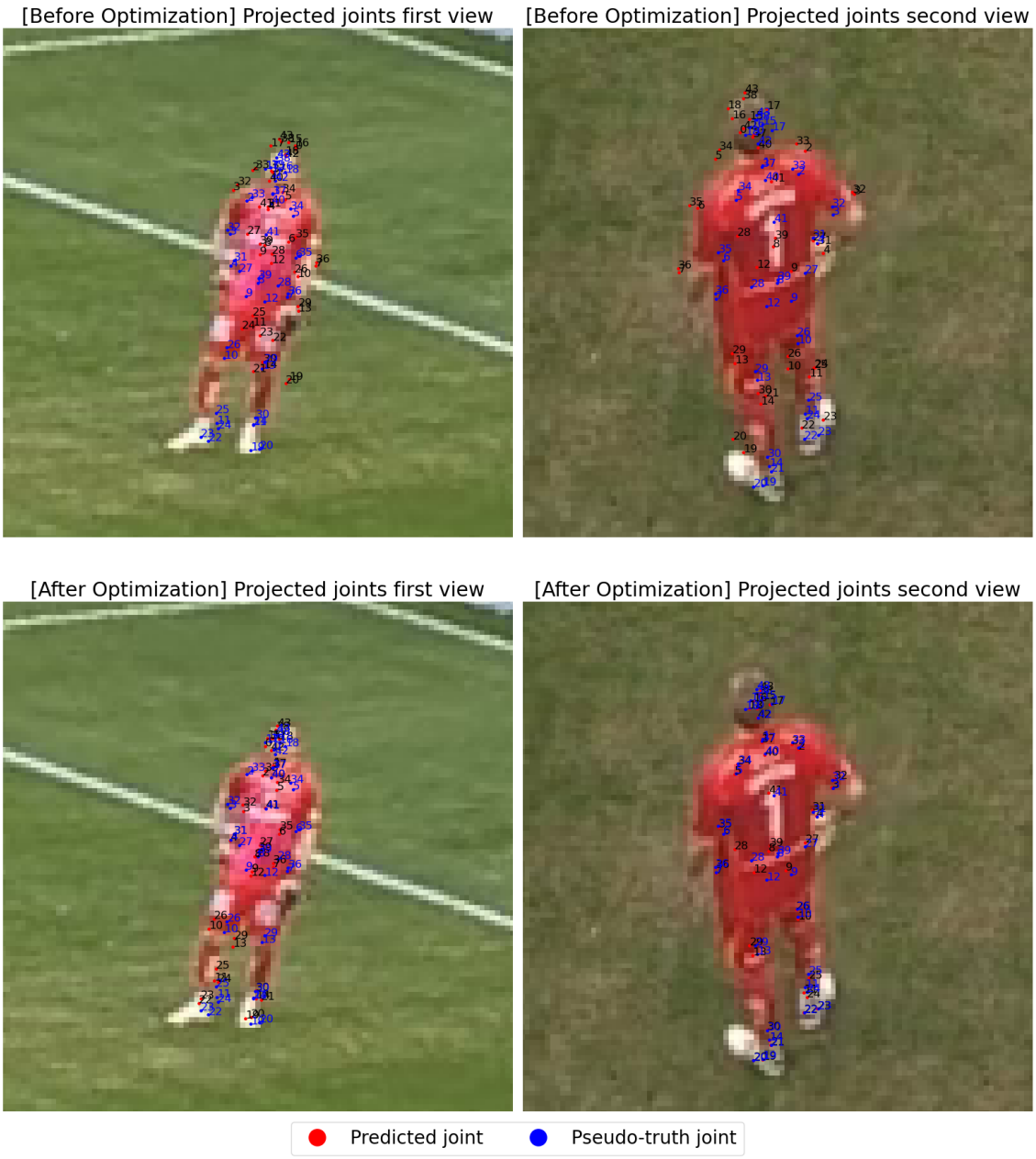


Figure 4.14: Visualization of the projected joints for the predicted SMPL pose and the pseudo-truth before and after optimization. Red dots represent the projected joints, while blue dots represent the pseudo-truth joints used during optimization. Each number corresponds to the ID of the respective joint.

In Fig. 4.15 and Fig. 4.16, the 3D SMPL pose is visualized before and after pose and location optimization. The poses are shown from five different angles to highlight their 3D characteristics. As seen in the figures, while the pose has undergone slight changes, it has become significantly more disfigured and no longer resembles a realistic human body.

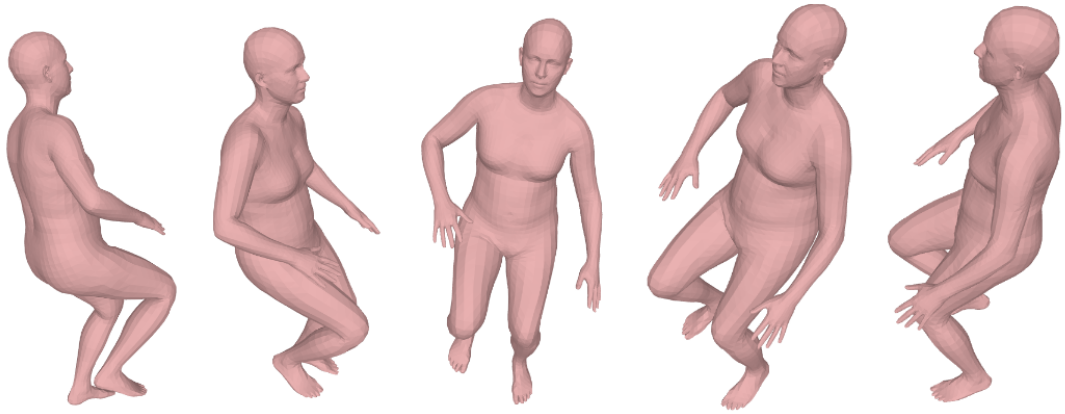


Figure 4.15: The predicted SMPL pose before pose and location optimization.

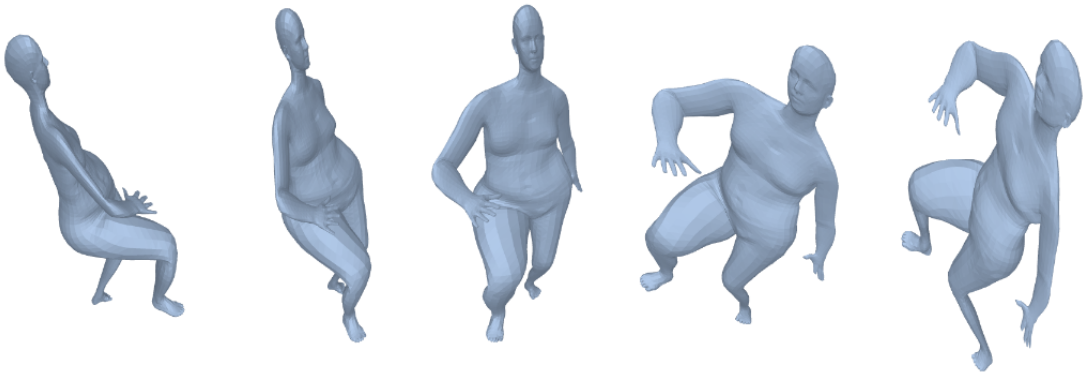
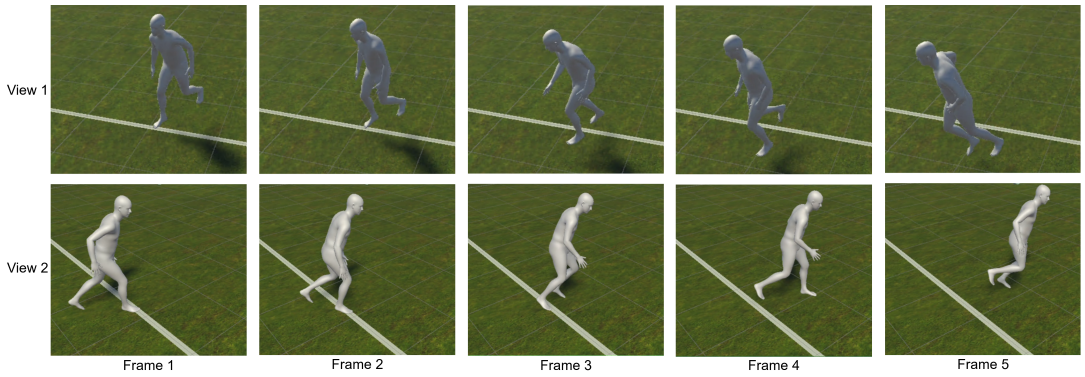


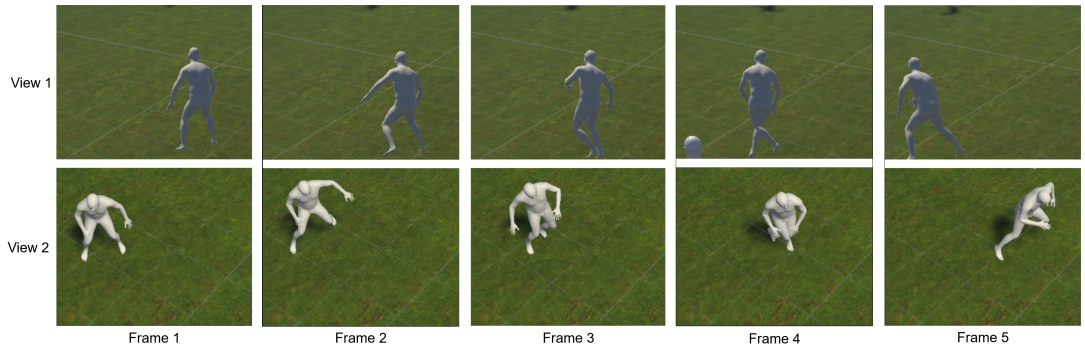
Figure 4.16: The SMPL pose after pose and location optimization.

4.5 Animated and Converted 3D Poses

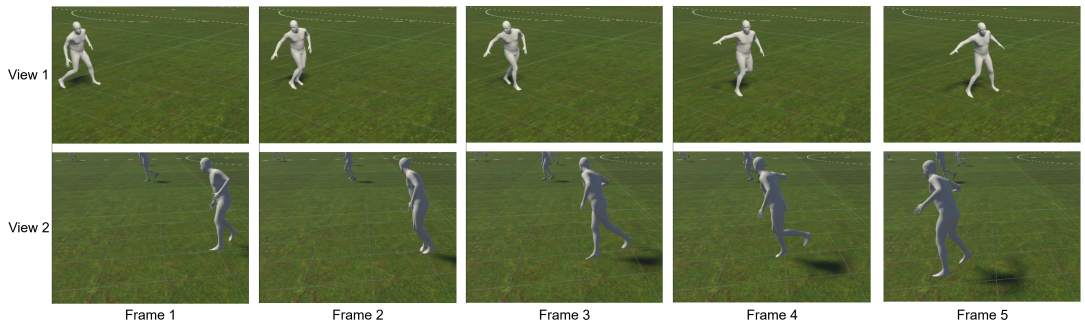
Results from the animation of player poses are presented as 5-frame snippets since video illustration is unavailable. Each animation is illustrated from two different views for every frame. In the plots, each row corresponds to a distinct view and the columns correspond to the different frames. The leftmost columns represent the starting poses of the animations and the rightmost columns represent the ending animations.



(a) Player 4 animated in 5 frames



(b) Player 6 animated in 5 frames



(c) Player 11 animated in 5 frames

Figure 4.17: Three animations from two different views seen inside the Unity game engine. The animated poses from Blender were exported as a `.gltf`-file to be rendered and visualized in Unity.

5

Discussion

This chapter explores the various results presented earlier, examining their significance for player localization and pose estimation in football. It looks at the accuracy of camera calibration, the precision of pose estimation, and the challenges in player localization. The goal is to highlight both the strengths and limitations of the explored methods. Additionally, the chapter discusses the performance of the multi-view pose and location optimization. Finally, suggestions for potential future work are also covered.

5.1 Camera Calibration

The results of the projections in Fig. 4.2 and Fig. 4.3 demonstrate that the camera calibration model is quite accurate, as the red dots align with the intended field lines. There are, however, slight inaccuracies, most notably along the center field line in Fig. 4.2, where the projected line does not perfectly align with the field line in the image.

For the purposes of player localization, this inaccuracy is acceptable since the projections are still nearly correct. However, this could pose a significant issue for the optimization of player poses and locations. The projection of joints for a player is quite sensitive to the estimated camera parameters, and even slight inaccuracies will result in incorrect joint projections. This means the projections will not align with the player in the bounding box, causing an offset. Consequently, this will corrupt the optimization process as it will attempt to optimize for an invalid pose.

The uncropped projections of the points, as seen in Fig. 4.4 and Fig. 4.5, show that the projections become highly unstable outside the image used for calibration. This is not an immediate issue since this project only handles points visible within the images. No poses or locations outside the visible area in the images are predicted or estimated. However, this would become a problem in scenarios where extrapolation of player positions is necessary when they move out of the original camera views.

5.2 Pose Estimation

The pose estimation results indicate that the projections of the estimated poses align well with the bounding boxes in simpler cases, as seen in Fig. 4.6. The general poses of the players are accurately captured, and the projected joints of the estimated SMPL model align well with the pseudo-truth.

However, there are some inaccuracies. For example, in player (a), the right foot and right hand are not correctly angled. The foot is noticeably misplaced, causing subsequent joint projections to be off. While the incorrect hand placement might not be critical, accurately predicting the feet is crucial for pose estimation in football. This issue is also visible in player (c), where the side view shows the player standing on their toes.

The more challenging cases, shown in Fig. 4.7, highlight inherent issues with the projected poses. Although the model captures the poses of occluded players (c) and (d) quite well, depth ambiguity becomes apparent. While the projected poses might seem correct, the side view reveals that player (d) is incorrectly tilted and player (c) has one leg in the air. This tilting issue is also evident in players (b), (e), and (f), where the poses project correctly but the side view shows the poses are not placed firmly on the ground as you expect from the images. Finally, player (a) shows the problem of visual ambiguity during occlusion, where the arm is covered by another player, leading to an unnatural estimation of the hand being straight out.

5.3 Player Localization

Localization and subsequent placement of the players' estimated poses on the virtual football field must be accurate to create visually appealing results. The results of positioning players using only monocular video, as shown in Fig. 4.8, fall short of the necessary accuracy for robust player localization.

The projection depths (green plots) show distinct noise, resulting in unrealistic or unlikely player movements in the 2D bird's-eye-view plots. This is especially clear in (c), where the player's movement follows a noticeable wavy pattern. This wavy pattern makes the player's movements on the synthetic football field unrealistic and visually unappealing in VR. In addition to these inadequate 2D positions, the hip heights are also notably noisy. The noise in hip heights may also contribute to the inaccuracies in depth estimations.

For these results, the projection of players along a ray until a given hip height assumes that the players are anchored to the ground. However, this is seldom the case as football players frequently walk, sprint, and jump. In the latter two, this naive projection does not work as intended. When a player is captured mid-jump or while sprinting, their target hip keypoint in the image plane changes slightly from the previous frame, resulting in a larger projected depth. This is because the ray calculated from the camera, based on the hip keypoint location in the image frame, shifts slightly upwards during sprinting or jumping.

Consequently, this upward shift alters the angle of the ray, leading to a larger projected depth calculation. This is because it relies solely on the hip height of the pose as the target height for projection, rather than accounting for the hip-to-ground height. This effect is likely a significant factor causing the wavy movement pattern visible in (a), (b), and (c).

To make the depth estimations from projection more accurate, it would be helpful to identify moments when players are planted (anchored) to the ground. The ability to detect anchored frames for every player would likely mitigate the resulting wavy movements. The results regarding detecting when players are anchored to the ground, as illustrated in Fig. 4.9, offer a promising approach for identifying instances when players' feet make contact with the ground.

A distinct pattern emerges across plots (a), (b), and (c), showcasing alternating speeds for each foot. It becomes apparent that while one foot displays rapid movement, the other tends to move slowly and often falls below the specified red threshold line. This observation highlights how effectively speed measurements capture moments when players' feet could be determined as anchored.

Nevertheless, a simple threshold speed does not provide robust ground detections, as shown in Fig. 4.10. The results for player 4 appear promising, with ground detections mostly correct despite some noise. In contrast, the plot for player 6 indicates that the movement threshold is set too low, anchoring only the player's right foot in frames 343 and 344. Conversely, for player 11, the threshold is overly strict, anchoring both feet in every frame, which is incorrect. It is visible that player 11 is walking away from the camera, so both feet should not be considered anchored in every frame.

Calculating the 3D location of a player in anchored frames often results in partially incomplete data, which was completed using linear interpolation. These results are illustrated in Fig. 4.11. The blue dots representing locations derived from linear interpolation appear mostly promising. The interpolated values in the 2D birds-eye-view plots appear smoother with less noise in both (a) and (b).

However, the bird's-eye-view plot in (c) shows that the movement remains noisy. Additionally, the linear interpolation seems to have exaggerated the unnatural player movements between frames 450 and 480, resulting in visibly unappealing outcomes. The overall movement and locations plotted in (c) still appear wavy and scattered. In addition, the illustrated results for interpolated hip heights seem robust in their interpolations, all the blue dots appear in reasonable hip height locations. However, this linear interpolation fails to account for the dynamics of walking or sprinting, where gravity continually affects the players.

Examining the results for the projected depths at ground detections displays a substantial amount of noise. Notably, there is almost no difference when comparing player 11 with and without ground detection, as shown in Fig. 4.11 and Fig. 4.8. The depth plots generated by the ground detection method show slightly sharper and less rounded peaks in both (a), (b), and (c), which was unexpected from this method.

The hypothesis was that the ground detection and subsequent depth estimation would result in less noise if a player were known to be positioned on the field rather than in mid-air. The expectation was that these refined estimations could then aid in interpolating any unknown player locations before or after these instances, thereby resulting in smoother and less noisy estimated 3D positions for every frame.

However, the actual results did not align with these expectations. This is likely due to the complexity of estimating depth for players far from the camera. Slight deviations of hip height or hip keypoint location in the image plane between frames lead to large differences in estimated depth. Moreover, the ground detection method occasionally underperforms in some scenarios, further exacerbating the error. In conclusion, it is evident that monocular depth estimation is a challenging task, and it is clear that additional information is necessary to improve its reliability.

Triangulating the 3D location from two views significantly improves the accuracy of depth 3D location estimations which is demonstrated in Fig. 4.12. The birds-eye-view plots all appear to capture the players' movements more realistically and accurately, with minimal noise remaining. The depth estimates in both (b) and (c) appear smooth and robust.

However, some noise is visible in plot (a) around frames 100 and 200. The noise around frame 200 is most likely due to inaccuracies of the triangulated location. This is reinforced when examining the hip height and bird's-eye-view plots, which show significant noise at frame 200, likely causing the observed depth noise. Overall, the results are promising and align with prior expectations, demonstrating the benefits of triangulation by using additional video from another angle. However, the extra requirement of an additional temporally synced camera angle should not be understated.

5.4 Pose and Location Optimization

The optimization of pose and location shows promising results. Fig. 4.13 plots the loss during the optimization and indicates that the process is indeed converging, with all losses decreasing and slowly plateauing. This trend is expected in successful optimization. Notably, the first view loss starts lower and finishes lower, which is anticipated since the pose estimated from this view is used as the initial starting point. Interestingly, the discriminator loss for the initial pose is also high, suggesting that the predicted pose from the first view is not initially considered realistic, which aligns with the need for further optimization.

Fig. 4.14 illustrates the expected pattern from the optimization of the projected joints. The projected joints after optimization are significantly closer to the pseudo-truth than before optimization. The projected points in the second camera view are also notably closer to the pseudo-truth points than in the first camera view, likely because the pose from the second camera view is used as the initial starting point.

However, when comparing the 3D SMPL pose before and after optimization, as shown in Fig. 4.15 and Fig. 4.16, it is clear that the post-optimization body shape is

severely deformed and no longer realistic in its body structure. While the actual pose of the model has been altered as expected, the limbs and body appear unnaturally shaped and unrealistic.

This deformation occurs even though the adversarial loss from the discriminator is decreasing, indicating that the discriminator loss has failed to ensure the body shape remains realistic. The discriminator is insufficient as a tool to maintain realistic body poses during optimization.

So, while the optimization process shows convergence and improves the alignment of projected joints with the pseudo-truth, it introduces significant deformation to the body shapes. This highlights the need for improved methods or additional constraints to ensure the optimized shapes remain realistic and anatomically correct.

5.5 Future Work

There are several promising possibilities for future work. One major area of improvement is the use of a more reliable and robust discriminator in pose and location optimization to ensure that the optimized poses remain realistic. Together with automatic player ID association between the two views, this would allow for automatic pose and position optimization of players on the field. This, in turn, would allow for the creation of comprehensive annotated datasets containing both pose and position annotations, provided temporally synced video from two different cameras is available. Such datasets would be valuable for training future models to predict both pose and location in a monocular camera setup.

Moreover, the ground detection results showed a promising alternating speed pattern for a player's feet, where one foot had high speed while the other had low speed, alternating as expected during movement. There is an opportunity to develop a more sophisticated model that utilizes this alternating pattern to detect whether a person is touching the ground. This could lead to more robust location estimations in a monocular camera setting.

Finally, the predicted SMPL poses suggest potential future work by incorporating temporal information from video into pose estimation or by using extended SMPL models with additional parameters specifically for controlling the feet. Incorporating temporal information could improve predicted poses in scenarios where players are occluded by leveraging predictable movement patterns. Using extended SMPL models with greater control over foot placement could improve the accuracy of placing feet on the ground.

6

Conclusion

The goal of this thesis was to explore how the poses and positions of football players could be estimated in a 3D environment using video of football matches as input. We propose a pipeline performing camera calibration, pose and location estimation, a possible optimization of both pose and location from two camera views, and conversion of the resulting poses and locations to a 3D environment.

We explored the SoccerSegCal model specifically tailored for camera calibration of football video. The model performs well on the football videos from IFK with some visible projective distortions. The distortions are significantly larger for 3D points not seen in the image.

The pretrained pose prediction model HMR2.0 was used to estimate the poses of the football players. It was generally successful in capturing the poses of the players, although it suffers from projective ambiguities when using monocular video.

Three main methods of locating players in a 3D environment were explored. The first two methods only used monocular video and suffered from inaccurate depth estimations. The final method we tested used two videos and triangulated locations for every player to improve the depth estimations. This triangulation showed great improvement over the previous two monocular-based methods.

A pose and location optimization using two different camera views was also proposed. Although it shows great promise, we found that it ultimately distorts the predicted poses too much to be viable with the current discriminator.

Finally, the last step of the proposed pipeline consisted of converting the predicted poses and locations to a 3D environment. This ultimately yielded an approach for converting annotated videos of football games to a representation of the game in a 3D environment.

We are satisfied with the results of the thesis. Currently, the available datasets for evaluating pose and position of players on a football field are limited. This has made it hard to evaluate the results gained from this project and a qualitative approach has been used. However, we believe this project provides a foundation to create such datasets in the future, especially if further improvements to the pose and location optimization are made.

Bibliography

[1] *Veo och SvFF i partnerskap*, sv, Nov. 2022. [Online]. Available: https://svff.svenskfotboll.se/nyheter/2022/11/veo-och-svff-i-partnerskap/ (visited on 05/29/2024).

[2] *SvFF och analys- och livestreamingtjänsten Spiideo i nytt samarbete*, sv, Aug. 2021. [Online]. Available: https://aktiva.svenskfotboll.se/nyheter/2021/08/SvFF-och-analys-och-livestreamingtjansten-Spiideo-i-nytt-samarbete/ (visited on 05/29/2024).

[3] M. Edlund and R. Gustafsson, *Showcasing the progress of the VR application to IFK Göteborg*, Feb. 2024.

[4] J. Humayun and H. Ganelius, "Tracking players and ball in football videos," M.S. thesis, Chalmers University of Technology, 2024.

[5] F. Anjou and A. Ekström, "Football Analysis in VR: Texture Estimation with Differentiable Rendering and Diffusion Models," M.S. thesis, Chalmers University of Technology, 2024.

[6] *IFK Göteborg*, sv-SE. [Online]. Available: https://ifkgoteborg.se/english/ (visited on 12/12/2023).

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

[8] S. Goel, G. Pavlakos, J. Rajasegaran, A. Kanazawa, and J. Malik, "Humans in 4d: Reconstructing and tracking humans with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023.

[9] S. Shin, J. Kim, E. Halilaj, and M. J. Black, "Wham: Reconstructing world-grounded humans with accurate 3d motion," *arXiv e-prints*, 2024. arXiv: 2312.07531.

[10] B. Wandt, M. Rudolph, P. Zell, H. Rhodin, and B. Rosenhahn, "Canonpose: Self-supervised monocular 3d human pose estimation in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021.

[11] R. Szeliski, *Computer Vision: Algorithms and Applications*, en. Springer International Publishing, 2022, ISBN: 978-3-030-34372-9.

[12] A. Cioppa, S. Giancola, V. Somers, *et al.*, "Soccernet 2023 challenges results," *arXiv e-prints*, 2023. arXiv: 2309.06006.

[13] *Spiideo/soccersegcal*, original-date: 2023-05-24T06:36:46Z, Mar. 2024. [Online]. Available: `https://github.com/Spiideo/soccersegcal` (visited on 04/28/2024).

[14] *Deeplabv3_resnet50 Torchvision main documentation*. [Online]. Available: `https://pytorch.org/vision/main/models/generated/torchvision.models.segmentation.deeplabv3_resnet50.html` (visited on 05/20/2024).

[15] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[16] Y. Tian, H. Zhang, Y. Liu, and L. Wang, "Recovering 3d human mesh from monocular images: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[17] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM Trans. Graph.*, Oct. 2015.

[18] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[21] R. Xiong, Y. Yang, D. He, *et al.*, "On layer normalization in the transformer architecture," in *Proceedings of the 37th International Conference on Machine Learning*, Jul. 2020.

[22] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[23] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[24] D. Mehta, H. Rhodin, D. Casas, *et al.*, "Monocular 3d human pose estimation in the wild using improved cnn supervision," in *2017 International Conference on 3D Vision (3DV)*, Oct. 2017.

[25] T. Lin, M. Maire, S. J. Belongie, *et al.*, "Microsoft COCO: common objects in context," *CoRR*, 2014. arXiv: `1405.0312`.

[26] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[27] A. Kanazawa, J. Y. Zhang, P. Felsen, and J. Malik, "Learning 3d human dynamics from video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[28] C. Gu, C. Sun, D. A. Ross, *et al.*, "Ava: A video dataset of spatio-temporally localized atomic visual actions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[29] J. Wu, H. Zheng, B. Zhao, *et al.*, "Ai challenger : A large-scale dataset for going deeper in image understanding," *arXiv e-prints*, 2017. arXiv: 1711.06475.

[30] B. Foundation, *Blender.org - Free and Open 3D Creation Software*. [Online]. Available: https://www.blender.org/ (visited on 05/24/2024).

[31] *BPY Python API Blender*. [Online]. Available: https://docs.blender.org/api/current/index.html#blender-4-1-python-api-documentation (visited on 05/24/2024).

[32] *Rotation Modes - Blender 4.1 Manual*. [Online]. Available: https://docs.blender.org/manual/en/latest/advanced/appendices/rotations.html#quaternion-mode (visited on 05/24/2024).

[33] *About FFmpeg*. [Online]. Available: https://ffmpeg.org/about.html (visited on 05/27/2024).

[34] T. I. F. A. B. (IFAB), *LAWS OF THE GAME 2023/24*, en-US. IFAB, 2023. [Online]. Available: https://www.theifab.com/ (visited on 06/11/2024).

[35] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019.

[36] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, eprint: 1412.6980, 2017.