ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Soccer Analysis: 3D Pose Estimation from Television Recordings

Bachelor Thesis

Tobias Buner

August 22, 2019

Advisors: Prof. Dr. M. Pollefeys, Dr. M. Oswald
Department of Computer Science, ETH Zürich

Abstract

The application of Computer Vision in the field of sports is increasing in numerous areas. The Swiss national soccer team approached the Computer Vision and Geometry Lab at the ETH to explore the possibilities of technologies to analyze soccer games using the existing TV cameras. In the following paper, we present a system that estimates three-dimensional human poses based on TV camera recordings and two-dimensional tracking data of the soccer game. The work is based on existing state-of-the-art algorithms for object detection and human 2D pose estimations. We present an Extended Kalman Filter to fuse the poses from different camera perspectives into world coordinates. The results confirm the feasibility of setting up such a system without large investments compared to other systems which are installed explicitly for this purpose. To achieve the desired accuracy, many points would have to be improved upon.

Contents

List of Figures

1 Introduction

The highest levels of competitive sports such as soccer, american football, tennis or the Olympic Games are among the most-watched television broadcasts. Relating to this, FIFA announced that more than half the world joined an official broadcast of the World Cup 2018 in Russia [8]. Due to new several technological advancements such as high resolution cameras and more computational power, it is not surprising that an interesting field of computer vision has evolved from them.

Tools have been developed for the large number of viewers to bring them closer to the action. Most of the tools are exclusively designed for this purpose in terms of camera perspectives and are not build on top of existing TV cameras. Intel True View end-to-end technology solution[12] for example makes it possible to render 360-degree replays to give spectators the feeling of standing on the pitch himself. Other tools like the Vizrt's sports graphics and analysis tools [31] provide data driven augmented reality, and additionally enable the application for advanced analysis.

Figure 1.1: Freekick Analysis and Visualization example from Vizrt's sports graphics and analysis tool.
Source: [31]

Advanced analysis through visual computing in sports is important for teams but also for the referee and the audience. The teams use these tools to improve their tactics based on generated statistics or to learn from past mis-

takes. Involved spectators are fascinated by facts from the statistics such as the longest distance traveled or top speed. But there is also a direct impact on the game. The simplest example is the video assistant referee (VAR), the VAR draws the referee's attention to questionable situations and gives him the opportunity to look at the scene again. Obviously this example does not include any Visual Computing and is simply a review of the camera recordings. Another example of computer vision based technology in sports is the Hawk-Eye ball tracking in tennis [21]. This allows the trajectory of a ball to be tracked purely from video in order to decide if the ball was in or out. Normally the line umpires are responsible to call the ball out. Through the Hawk-Eye system, however, there is a new opportunity for the players to challenge the call of the umpire if they think the ball landed in. This challenge is then performed using the Hawk-Eye system and the result is displayed on the screens in the stadium to be transparent to the viewers.

1.1 Focus of this Work

The Swiss national soccer team approached the computer science department in order to explore state-of-the-art computer vision and visualization technology to analyze soccer games to generate and visualize player statistics and new performance measures of players or teams. The goal of this thesis was to extend the existing two-dimensional tracking system to three dimensions. In contrast to other existing tools in this area, the motivation was to aim for a systems which uses the existing TV cameras as source and not exclusively for this purpose installed cameras. This would allow new possibilities: if, for example, not only the position on the pitch is known for each player, but also a three-dimensional skeleton model, it would be possible to make a estimate of the field of view of the respective player.

Thesis Structure

Chapter 2 provides an overview of other, related work and discusses the differences and similarities to our approach. Afterwards the method of this thesis is explained in detail, with a short overview in the next section. The different results are presented and described in chapter 4 and at the end, there are some conclusions and what could be further improved in the work.

Outlook on the Method

The video material from multiple TV cameras which pan and zoom during the match and the two-dimensional tracking data were available. In chapter 3 each step is explained in more detail, but to provide a rough overview:

- In order to fit skeleton models to all players in 3D space, stable skeleton joint positions in screen coordinates had to be estimated with *OpenPose*[23].
- Camera calibrations were additionally necessary in order to be able to fuse the twp-dimensional positions.
- For the multi-view aggregation an Extended Kalman Filter was used.

2 Related Work

Various systems in this area exist already, such as, the Intel True View end-to-end technology solution [12] that is able to render a complete 360-degree replay. The technology behind it is roughly explained in video [13] as follows: There are about 38 high-definition 5K cameras installed in a stadium processing terabytes of data every minute across 38 different servers. The cameras generate not just images of two-dimensional pixels, but additionally they generate massive amount of volumetric data (voxels), pixels with volume, which allows them to capture height, width and depth. This means not to lose the depth when capturing a moment with the camera which allows the mapping of events in three dimensional space. In the figure 2.1, you can see how the 38 cameras are distributed in the stadium to capture the events on the pitch from every angle.

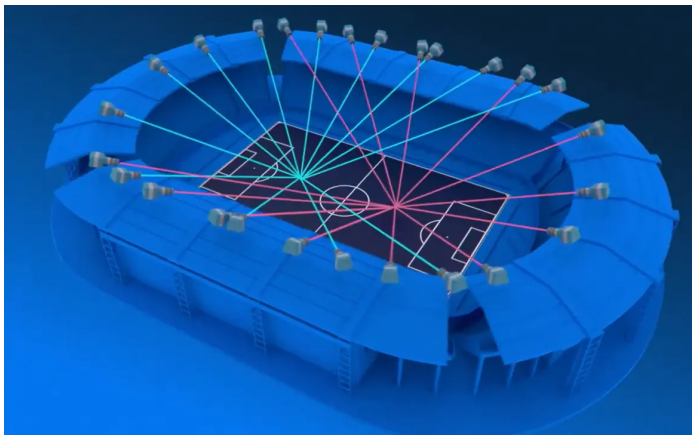


Figure 2.1: Location of the High-definiton 5K cameras for the Intel True View technology solution in a soccer stadium.
Source: [30]

The Swiss national soccer team asked to explore state-of-the-art computer vision and visualization technologies for the data of some TV cameras rather than installing new, expensive high-definition cameras around the stadium to track the bodies of the players. A large related research area has been erased in the last few decades on motion capture techniques.

2.1 Motion Capture

Motion capture tries to track and record the motions of the human body: the human body is interpreted as a system of rigid links connected by joints as expressed by Xsens[37], which is a leading innovator in 3D motion tracking technologies. The developed methods can be divided into different categories, but most technologies do not only benefit from the advantages of one category, rather they combine several techniques to improve the general outcome. The main distinction that is particularly relevant for this work is the difference between marker-based motion capture and markerless motion capture [22].

Marker-Based Motion Capture

Marker-Based Motion Capture require to wear some form of markers on the body. These markers can be further categorized into mechanical, magnetic, acoustic and optical [37]. Even the markers are further categorized into active and passive markers. For example, passive markers reflect the light back to the cameras. In contrast, the active markers are equipped with a battery and emit their own light [22].

- Mechanical motion systems directly track the joints angles using a exoskeleton worn by the person [33]. This technology is very popular in the film industry.
- Magnetic motion capture systems use the markers as sensors to measure low-frequency magnetic fields generated by a transmitter source. The sensors measure the strength of these magnetic fields in order to calculate positions and orientations [20, 33, 37]. Magnetic systems do not suffer from line of sight problems, thereby the method is useful if some body parts are concealed by others.
- Acoustic tracking systems use ultrasonic pulses and can compute the position through the time-of-flight of the pulses with triangulation. To work reasonably, a clear line of sight is needed and to avoid reflections this method is most suited to outdoor use [37].
- Optical motion capture uses multiple cameras to track the markers which can be passive or active for this purpose as described above [37]. To extract the markers, 3D position triangulation is used just as in the acoustic system. There are even single camera systems but these require additional sensors to measure the lost depth in the image when triangulation is not possible [25].

In general, those approaches are all limited to the fact that the athletes are required to wear some form of markers on their body. In the context of soccer it would be possible to place the markers inside the clothes and shoes,

but that would have to be approved by the FIFA for official games and also by the players themselves. However, exactly these technologies are used for video games like the FIFA video game series to capture the movements of the players more realistically.

Figure 2.2: Messi in Xsens' motion capture suit for FIFA 16 to animate his dribbling style as realistically as possible.
Source: [36]

Markerless Motion Capture

Due to the numerous researches in the field of computer vision, markerless optical motion detection systems have been made possible. These systems do not rely on a suit with markers which has to be worn, instead they rely on computer vision algorithms to reconstruct the 3D motion from videos and sensors [25]. An example for such a system is Microsoft's Kinect [28],

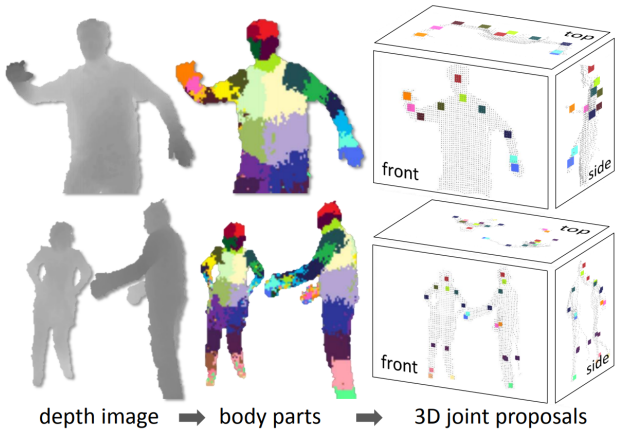


Figure 2.3: Overview for the 3D positions reconstruction from one single depth image.
Source: [28]

they localize the 3D positions of body joints from one single depth image. Roughly described, they designed an intermediate body parts representation based on an object recognition approach. This presentation consists of several localized body part labels which cover the whole body. This intermediate representation simplifies the difficult pose estimation problem into a simpler per pixel classification problem. Fur further details refer to [28].

2.2 Human Poses Estimation

During the last few years, the human 2D pose estimation has been greatly improved. Through better object detection algorithms, such as the one presented in the section 2.3.1, common approaches try to detect a person and run a single-person pose estimation for each detection. The runtime of these so called Top-Down approaches is proportional to the number of detected people in the image [2]. To avoid this complexity, so called buttom-up approaches have been developed as *DeepCut* [24] or *OpenPose* [2]. A rough overview to how they work is given in the section 2.3.3 for the *OpenPose* approach.

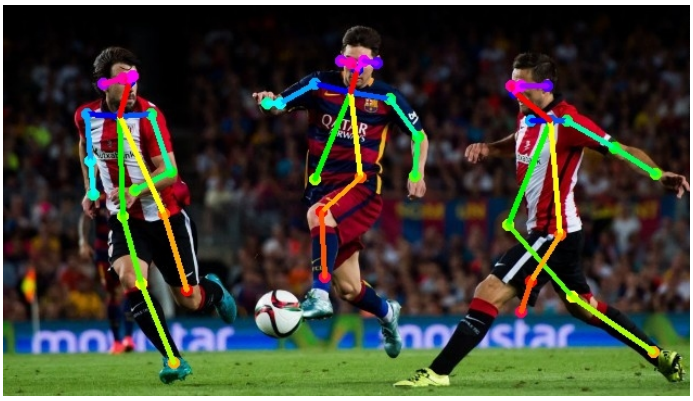


Figure 2.4: Human 2D pose estimation example with OpenPose.
Source: [3]

The resulting 2D pose estimation is shown in the figure 2.4. With these technologies, we were able to estimate human body joints on two dimensional images which is the foundation for 3D human pose estimations.

In order to understand the 3D geometry, the depth that is needed is exactly what is lost in 2D images. To recover this depth from images or videos, there are already comprehensive work such as [11, 7]. But these are applied to normal scenes and not explicitly to human bodies. The article [1] describes a method for automatic 3D human pose and shape estimation from a single image. This approach builds upon the 2D joints results of *DeepCut*[24].

Afterwards they fit a statistical body shape model to the 2D joints. The corresponding model is called the *Skinned Multi-Person Linear Model (SMPL)* [19] which is a realistic 3D model of the human body learned from thousands of 3D body scans. To fit the model to the 2D joints they minimize an objective function including the error from the projected 3D model joints and the previously estimated 2D joints. In the figure 2.5, an example result from the automatic 3D human pose and shape estimation is presented.

Figure 2.5: To the 2D joints of the original image (not shown), a *SMLP* model is fitted as seen in the middle and on the right side the resulting 3D model is rendered from a different viewpoint. **Source:** [1]

In contrast to our method they fit a complex statistical body shape model to the 2D joints and end up with a 3D shape model. The three dimensional joint positions are sufficient for our purpose. Additionally we have access to multiple camera views which enable triangulation based methods. Additionally, we can benefit from several consecutive frames instead of looking at each frame individually.

2.3 Soccer on Your Tabletop

This section covers the project *Soccer on Your Tabletop*[26], which provides a moving 3D reconstruction of a soccer game from a single monocular video. The estimation of the depth map of each player represents the core of their work. In the figure 2.6, a rough overview of their pipeline is given, with the first few steps explained in more detail in the following sections. The reason for the more detailed explanation is their integration in our work which is further outlined in the method chapter.

The input are frames from a single monocular video that is being processed. The player analysis consists of several building blocks of existing work. In the first step a state-of-the-art object detection algorithm called *Detectron*[10] is used to detect the people in each frame. The 2D poses for the recog-

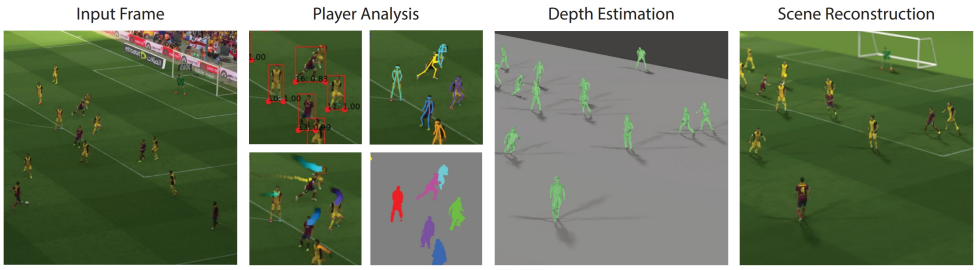


Figure 2.6: Overview of their reconstruction pipeline. In the first step the players from the input frame are analyzed and the camera calibrated. Through the player analysis they segment the player and reconstruct the depth map on the pitch per player. At the end they render the scene in 3D for a complete scene reconstruction.
Source: [26]

nized people are then determined using *OpenPose*[2]. These poses together with the detection are then used to refine the people segmentation [38]. For the per player depth map on the pitch, they developed a deep neural network. The network was trained on video game data, to be more precise they extracted depth maps from EA FIFA games to train the network. The advantage of this training data is the specific form in terms of movement, clothing and camera perspective [26].

Building Blocks

As mentioned above, there are multiple building blocks which are also part of our method. For this purpose the theory behind them are further explained in the next few sections. How they are integrated into our work is then explained in the method chapter.

2.3.1 Detectron

Detectron[10] is a system developed by Facebook AI Research and implements multiple state-of-the-art object detection algorithms such as Mask R-CNN [27] and RetinaNet [18] among others. They provide a subroutine to recognize objects on a input image.

Soccer on your Tabletop modified the originally subroutine provided by *Detectron* which outputs the visualization of the detected objects in PDF format. Instead, bounding boxes around the recognized objects and segmentation masks are output. At this point in time not only the players but also the spectators and the ball were recognized. In further processing, the objects outside of the pitch get removed using the camera calibration. Additionally, the bounding boxes with an area below a certain threshold are removed just to keep the boxes for the players [26]. These are used in later processes.

2.3.2 Calibration

This section covers how *Soccer on your Tabletop* estimates the camera calibration. During this step, the camera matrix is determined for each frame of a camera. The camera matrix M transforms homogeneous 3D world coordinates to homogeneous 2D screen coordinates and can be decomposed into an intrinsic and extrinsic camera matrix which is described in more detail in section 3.1.

Manual Initialization

The calibration step consists of two phases, during the first one, the algorithm requires to select manually four pairs of matching points. One pair includes a point in the first frame of the calibrated video and the corresponding reference point where it belongs to in a two-dimensional soccer field. Due to the fact that the soccer fields is in the x-y plane we have three-dimensional coordinates for the reference image with z-coordinates equal to zero. While the selected point on the frame consists of two-dimensional image screen coordinates. With these point pairs, the intrinsic camera matrix is determined by a grid search. The grid search estimates for different focal lengths the camera matrices. Afterwards, the 3D coordinates are projected into the first frame using the previously estimated camera matrices in order to compute a score for the corresponding focal lengths in a least-squares fashion. The result of the grid search are the focal lengths with smallest score and thereby the projection with the smallest error. For the principal point offsets the height and width of the image were simply divided by two under the assumption that the principal axis intersects the image at the center.

The extrinsic camera matrices are then estimated using the matching points from before with the estimated intrinsic matrix. This step works in the same fashion as the grid search to find a solution which minimizes the reprojection error. The use of the *Random sample consensus (RANSAC)*[34] scheme, a commonly used iterative method in computer vision which estimates the model parameters using observational data, makes the function robust to outliers [4].

Calibration Propagation

After the manual initialization for the first frame, in the second phase the camera matrices for the next frame are estimated using the matrices from the previous frame. First the edges in the new frame are detected using the canny edge detector [5]. With the use of the Sobel kernel, the image is filtered in order to get the edge gradient and direction for each pixel. Afterwards, through non-maximum suppression all pixels which are not

the local maximum in the direction of the gradient are removed in order to consider only pixels which are part of an edge. Hysteresis Thresholding classifies all edges according to two thresholds. Those between the two thresholds are the critical ones and are classified based on their connectivity [5].

The edges are used to obtain a representation of the image where the pixel values indicate the distance to the nearest background pixel [6]. By knowing the camera matrices for the last frame, the soccer field is drawn on the image. This in turn is unprojected to obtain a synthetic 3D field. Along with the image with the distance to the nearest background pixel, an objective function is defined and minimized to obtain the new camera matrices for the actual frame. The python code is provided by the *Soccer on your Tabletop* [26] project.

2.3.3 OpenPose

Through *Detectron* the location of recognized objects were given but the human 2D poses were needed. To solve this problem, the realtime multi-person 2D pose estimation *OpenPose*[2] was used. The systems outputs an array of people objects including the poses using the *JSON* file format.

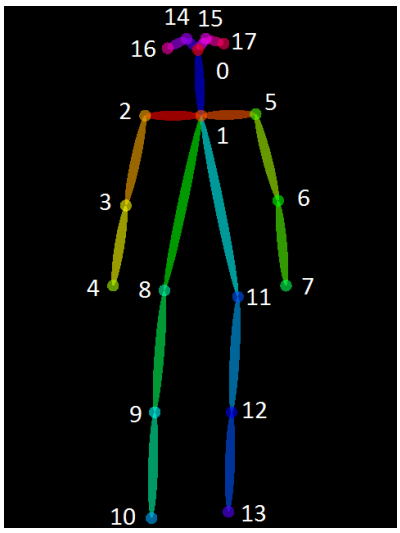


Figure 2.7: Numbering of the 18 keypoint of the COCO pose format from *OpenPose*.
Source: [23]

The pose output format (COCO) was used consisting of 18 keypoints as shown in figure 2.7. In addition to the corresponding keypoint screen coordinates, the output also includes so called confidence scores for each keypoint, in the range of $[0, 1]$, to express their validity.

OpenPose Algortihm

This section is intended to give you a rough understanding of their algorithm as described in their paper [2]. Common approaches estimate the poses per person but *OpenPose* follows the bottom-up approach in order to avoid runtime complexity proportional to the number of people. So the algorithm predicts body part locations independent of the number of people in the image and tries to connect the individual body parts so that the results are people. This is the motivation for the so called bottom-up approach, but in contrast to other methods such as *DeepCut*[24] the final analysis to map the body parts together is more efficient.

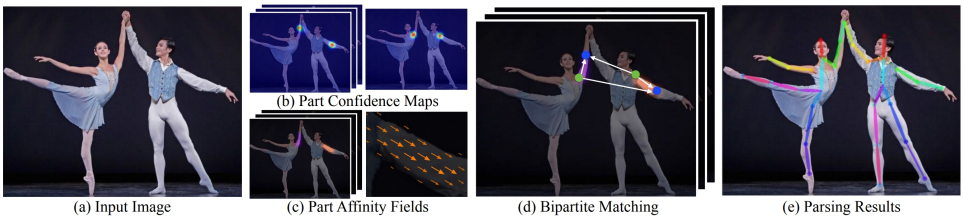


Figure 2.8: *OpenPose* Pipeline Overview. The input image (a) is processed to produce the confidence maps (b) and part affinity fields (c) using a CNN. The result is formulated as a bipartite matching problem (d) in order to solve the part association (e).
Source: [2]

At the beginning, the input image is processed and analyzed using a convolutional neural network (CNN). The generated feature maps are passed into the first stage, which produces a set of part affinity fields (PAFs). A 2D vector in each pixel of every PAF encodes the position and orientation of the limb which is not truly a human limb but a connection of two body parts as shown in figure 2.8 (c). In the following stages, the predictions from the previous stage and the original feature are used to refine the estimated PAFs. Afterwards, the process is repeated in the same fashion for the body part location to refine the confidence maps. A confidence map is constructed for each body part as shown in figure 2.8 (b) for the left elbow and left shoulder. For each pixel there is a probability that the corresponding body part is located there which is stored in these confidence maps. So for a single person each confidence map should have one peak for the corresponding location of the body part in the image. These steps in the network differ from other approaches where both the PAFs and the body part locations are refined together but the refinement of the confidence map shows no effect on the PAFs refinement and therefore they are separated, which leads to performance improvements and better accuracy. The resulting confidence map for each body part of the network is a collection of all the individual confidence maps for this body part which are combined through non-maxmimum sup-

pression to avoid averaging along the different peaks and therefore be able to distinguish between equal body parts that are close to each other.

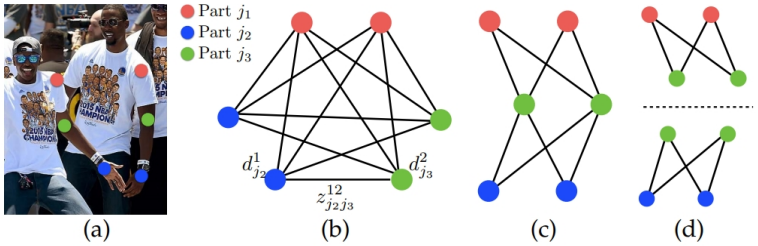


Figure 2.9: *OpenPose's* Part Association formulated as a matching problem with the use of part affinity fields. (a) shows the detected body parts which are then connected in a complete graph (b). The first relaxation reduces the graph to spanning tree skeletons (c) and the second relaxation decompose the graph into bipartit matching subproblems (d).
Source: [2]

Given the detected body parts derived from the confidence map for an unknown number of people, it is now necessary to map them together to full body poses. For this purpose a confidence score for each pair of detected body parts (candidate limbs) is computed using the line integral on the PAF along the connection between these two parts. Remember the PAFs preserve location and orientation that contribute the additional information needed for this step. Figure 2.9 (b) shows how each node represents a body part where the color corresponds to the type of body part and the edges express the so called candidate limbs weighted by the confidence score. The part association problem is reformulated as a matching problem which is known to be NP-Hard. But in order to solve the matching problem efficiently, two relaxations are described. The first relaxation reduces the number of edges to obtain spanning tree skeletons as presented in figure 2.9 (c), while the second relaxation further decomposes the graph into bipartit matching subproblems (d). The matching of these subproblems of adjacent tree nodes can then be determined efficiently and independently.

2.3.4 Depth Map Estimation

This section describes the key component of their algorithm, how to estimate the depth maps of the players for a monocular video. In our case we have access to multiple perspectives, that allows us to use other techniques to estimate the depths of the players. But to give you a rough overview how they solved this problem with a single image the key concepts from their paper [26] are explained here.

As mentioned above, they trained a deep neural network with ideal training data (image and depth map pairs) from EA FIFA games. This allows to

train the network explicit for their purpose, namely football player. The data contains players in various poses with the appropriate clothes from a typical camera perspective of soccer games. Access to the GPU allows them to capture the images and depths of the game which are then further processed for the resulting image-depth pairs. These are used to train the neural network in order to estimate the depth of a soccer player in new unseen images. The input for the network are simply the crops around the player together with the segmentation mask of the player [26].

2.4 Filterpy

FilterPy is a Python library that implements a number of Bayesian filters including an Extended Kalman Filter. The author of the library also wrote a book titled *Kalman and Bayesian Filter in Python* [15]. Because the book was written as an Ipython Notebook, it offers an interactive guide to deal with Bayesian filters. It starts with some simple filters up to the Kalman filter. It describes the implementation of the library through many examples and explains the mathematical background. For more information visit the the GitHub page [17].

3 Method

This chapter describes how the method is designed and explains each step in more detail. The input consists of the input frames for all used cameras and a file with the two dimensional tracking data of the players on the pitch. All the data was provided by the swiss national soccer team. The building blocks of our method are executed sequentially due to the reason that the results are needed for the next stage. In the figure 3.1 there are two areas, the gray and blue one. The difference betweend them is that, during the gray phase, the input images are processed per camera and independent of the other camera views. This phase includes the camera calibration and player

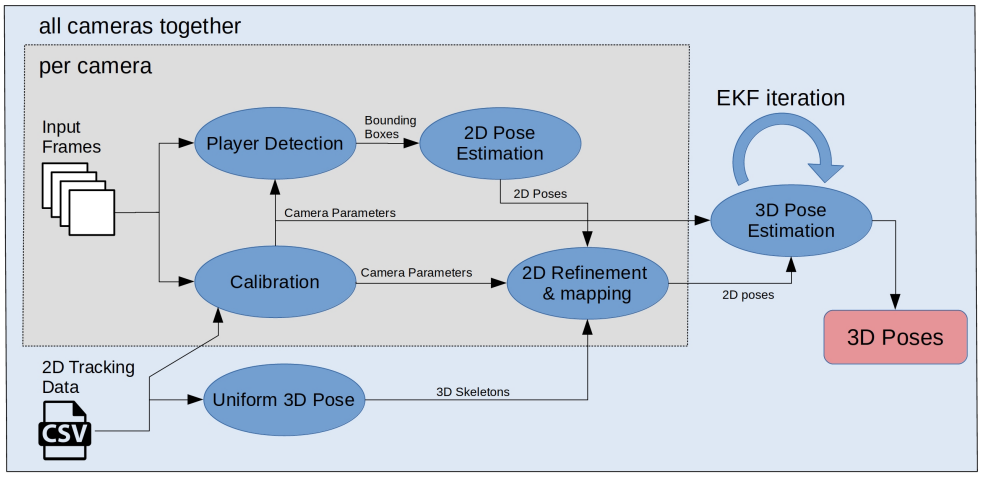


Figure 3.1: Overview of the workflow. Each step gets described in detail during this chapter. The gray area is the process for each camera separately while in the blue area the different cameras are merged.

detection. Afterwards the 2D poses are estimated and refined with the use of the camera calibration. In order to map the estimated poses to real players, the 2D tracking system comes into play and helps to identify the player. The tracking informations are camera independent, so they remain the same for all cameras. Until this step, the generated 2D poses for each camera forms the input for the blue phase. The 3D pose estimation is performed using an Extended Kalman Filter (EKF) and the results are the three dimensional keypoints of each player for every frame.

Several camera recordings were available, but some cameras had fisheye

lenses that we could not calibrate properly due to the fact of the strong visual distortion. The pipeline works for a variable number of cameras, this is particularly relevant for the EKF to estimate the 3D poses. Therefore the EKF is written for a variable number of cameras. The results were produced using two or three different camera views, but more on this in the chapter 4. The cameras are called K8, K1 and K9 and the first frame is shown in figure 3.2.

Figure 3.2: First frame of the cameras K8, K1 and K9.

Frame Extraction and Synchronization

During the entire process, the input videos were processed in single frames, so we extracted the frames at the lowest frame rate of the videos which was 25 fps. This leads to the time step $\Delta t = \frac{1}{25 fps} = 0.04s$ between two consecutive frames which is used in later steps. Additionally, for later purposes the different camera frames must be synchronized. This step was performed manually for the start of the game where the first players touches the ball.

3.1 Calibration

In order to work in three-dimensional space it is required to know how the pixels in a two-dimensional image are related to the three-dimensional world. For the camera model we consider the pinhole camera model for simplicity despite the fact that some cameras suffer from distortion. As mentioned in section 2.3.2, we were able to calibrate the cameras and got the corresponding camera matrix $\mathbf{M}$ which transforms homogeneous 3D world coordinates to homogeneous 2D image coordinates. The calibration for each frame is saved for later use.

Camera Matrix

As already mentioned the camera matrix **M** can be decomposed into an intrinsic (**A**) and extrinsic (**R** and **T**) matrix:

$$\mathbf{M} = \mathbf{A} \left[\begin{array}{c|c} \mathbf{R} & \mathbf{T} \end{array} \right] = \begin{bmatrix} f_x & \gamma & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \end{bmatrix} \tag{3.1}$$

R and **T** together build the extrinsic camera matrix which describes the camera's position and view direction in world coordinates. In order to apply the extrinsic matrix to homogeneous 3D world coordinates, it is common to extend the matrix by an extra row $[0, 0, 0, 1]$. The extrinsic matrix transforms the world coordinates by a rotation matrix **R** and a translation vector **T** to the 3D camera coordinate system.

The intrinsic camera matrix **A** transforms the 3D camera coordinates to 2D homogeneous screen coordinates. The matrix consists of:

- Focal length f_x, f_y. These describe the distance between the image plane and the pinhole of the camera.
- Principal point offsets x_0, y_0. The perpendicular line to the image plane which intersects with the pinhole is called the principal axis. The intersection of the principal axis with the image plane is the principal point and the offsets describe this point with regard to the screen origin[29].
- Axis skew γ, which is responsible for the shear distortion. For a true pinhole camera this is equal to zero[29].

The rotation matrix **R** and the translation vector **T** describe how the world is transformed relative to the camera. In order to obtain the location we have:

$$0 = \mathbf{R} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \mathbf{T} \tag{3.2}$$

$$\Rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} = -\mathbf{R}^\intercal \mathbf{T} \tag{3.3}$$

For the orientation of the camera we assume that the camera looks down the z-axis in the camera coordinate system as it is usually the case and can then extract the camera orientation as follows:

$$\begin{bmatrix} x_{direction} \\ y_{direction} \\ z_{direction} \end{bmatrix} = \mathbf{R}^\intercal \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.4}$$

Coordinate System

We determined the camera location in world coordinates, but where is the origin of our world coordinate system? The origin is set to the center of the pitch with the x-axis in the length and the y-axis in the width of the pitch. While the z-axis describes the height. For the dimension of the pitch we used the 2D tracking data provided by the Swiss national soccer team which is a metric system. The tracking data includes the two dimensional position of the ball where the data is only recorded when the ball is in the pitch. Therefore we extracted the maximum and minimum x and y coordinates of the ball to get the pitch size under the assumption that the ball goes out at least once on each side line. The cameras were calibrated with respect to this world coordinate system. Figure 3.3 shows each camera location and the corresponding direction of view, together with figure 3.2 they're giving a good overview to what the setup in terms of cameras perspectives looks like.

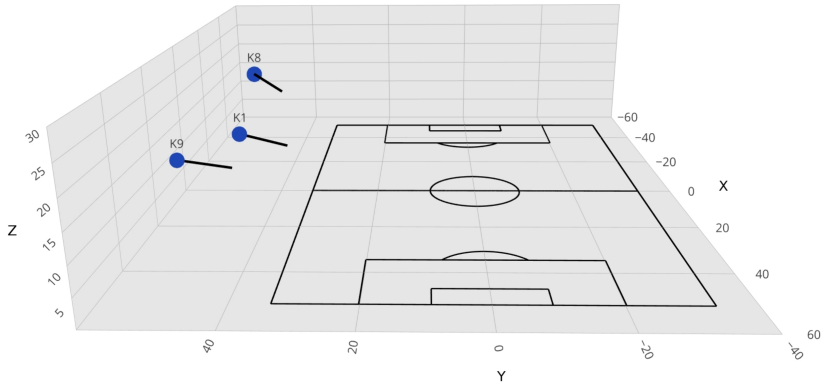


Figure 3.3: Location and view direction of the three cameras (K1, K8 and K9) next to the soccer field.

Improvement

The subroutine provided by *Soccer on your Tabletop* to calibrate the camera was improved in another project from a student at the same time. This was used for the camera K8 and K9, for the central camera K1 this approach did not work. We assume that the reason for this is the lower angle perspective, making it harder to detect the side line in the back of the pitch, which includes a lot of information about the orientation.

3.2 Player Detection

The player detection forms the first step to estimate the human 2D poses. With the use of an object detection algorithm[10], the segmentation masks were estimated as shown on the left side in figure 3.4. As mentioned in section 2.3 we can remove the irrelevant detections like the spectators with the help of the camera calibration. To do this, the soccer field is projected onto the image in order to classify if an object is inside or outside of the pitch. For the remaining segmentation mask the bounding boxes are constructed

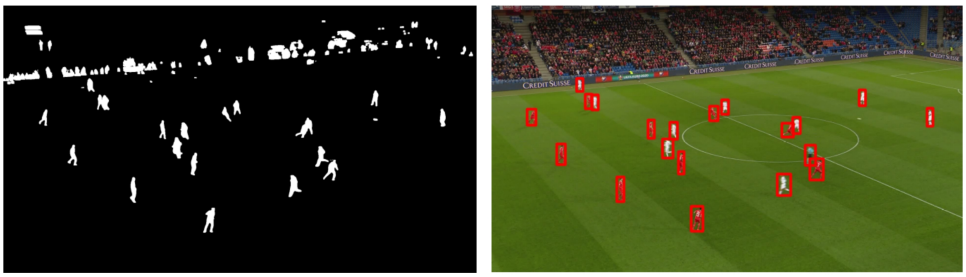


Figure 3.4: Detectron Output. On the left side the segmentation masks are shown and on the right side the refined bounding boxes.

which are later used for the player pose estimation. To refine the created boxes, those with an area below a certain threshold were removed to exclude possible small objects such as the ball. The result are the bounding boxes displayed on the right side of figure 3.4.

3.3 2D Pose Estimation

In this step the human 2D poses in screen coordinates were estimated using *OpenPose*[23]. The algorithm is designed for multi-person 2D pose estimation, but in most scenarios where the algorithm is applied, the camera is close to the ground and the distance to people is small. If the algorithm is used to estimate the human poses on the original frames, the results are not stunning. For players in the distant area of the pitch there are rarely estimated poses because the players were too small. At this point the previous defined boundig boxes come into play. In contrast to estimate the multi-person 2D poses on the entire scene the poses are estimated on crops of the frame. So for each bounding box a temporary image is cut out with the bounding box in the middle of it as demonstrated in figure 3.5.
Afterwards the *OpenPose* algorithm estimates the poses on this crop. This procedure allows to recognize more poses then on the entire scene. The big disadvantage is of course the loss of the computational efficiency of a real-

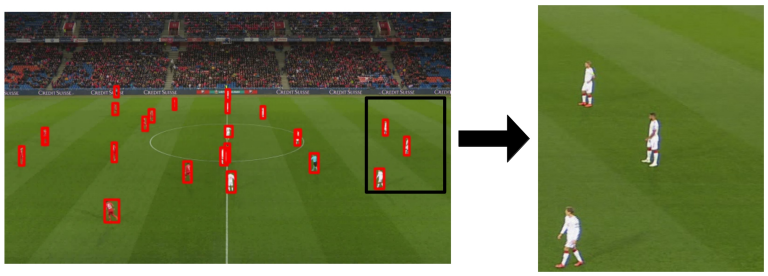


Figure 3.5: For each bounding box a part of the original pictures is cut out to estimate the 2D poses.

time multi-person 2D pose estimation when running the algorithm for 22 crops of one frame, but the goal is not to develop a method that runs in real time and the better results are worth the additional computational costs.

The screen coordinates of the estimated keypoints are then adapted to fit to the entire frame and saved for later use. An example result of the obtained keypoints is shown in figure 3.6. But the keypoints are not perfect, in fact there were several points to improve which is described in the next section. But in general we were able to estimate the poses of most of the players on the pitch.

Figure 3.6: Example result for the estimated human 2D poses.

3.4 2D Refinement and Mapping

As stated above, the generated 2D pose estimations are now refined in the same way as in *Soccer on your Tabletop* [26]. During this process the keypoints

for each person are tested to decide whether to keep this person's estimated keypoints. Remember that the pose output format (COCO) consisting of 18 keypoints was used. Three conditions are tested which must all be met to be maintained:

- The number of keypoints with a confidence score unequal to zero must be at least 8.
- The sum of all confidence scores must exceed a given threshold (set to 0.4).
- The confidence score of the neck keypoint (number 1 in figure 2.7) must be greater than a given threshold (set to 0.4).

Additionally a neck non-maximal-suppression algorithm[9], adopted from [26], is performed on all the neck keypoints of a frame to remove poses from the same persons derived from different crops. At the end, we test again if the poses are on the pitch and if not we discard them. The reason to test this again after the boxes got tested is the big crop area, where people outside the field can also be recognized. The resulting poses are the ones which are used in and referred to in later steps. In order to relate the poses for the same player from different camera views we need additional information about the estimated poses.

Mapping

In order to fuse the human 2D poses in screen coordinates to 3D poses in world coordinates as explained in the next section, we need some more information to map the estimated 2D poses to the corresponding player. For this purpose we take the additional 2D tracking data provided by the swiss national soccer team. This way we know the x, y position for each player on the plane of the pitch. Afterwards we generate so called uniform skeletons of the same size regardless of the player size and place them at the correct location on the pitch as shown in figure 3.7. These 3D skeletons are used to map our undefined 2D poses to a specific player. The mapping is performed for each pose separately. So for a pose we project each uniform 3D skeleton onto the image using the camera matrices to obtain their screen coordinates (x_s, y_s). In order to find the associated skeleton for the given pose, we sum up all the distances between each keypoint (subscript k) with confidence score (c_i) unequal to zero from the pose and the corresponding skeleton keypoint (subscript s) and divide the sum by the number of used keypoints to determine the skeleton with the smallest distance:

$$\arg\min_{skeleton} \left(\frac{1}{|\{c_i \neq 0\}|} \sum_{\substack{i=0 \\ c_i \neq 0}}^{17} \sqrt{(x_{k,i} - x_{s,i})^2 + (y_{k,i} - y_{s,i})^2} \right) \tag{3.5}$$

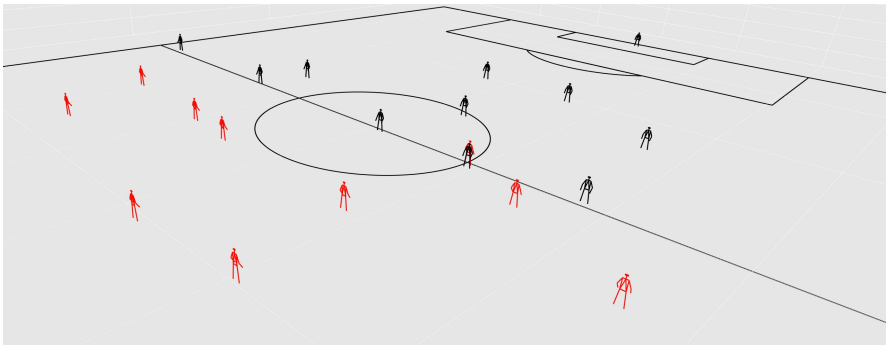


Figure 3.7: Uniform 3D skeletons initialization of the players with the use of the 2D tracking data.

This mapping allows us to combine the information from multiple camera views for the same event on the soccer field in the next step because we have the human 2D poses for the same players from different perspectives. These structured two dimensional poses for each frame serve as measurement values for the Kalman Filter which is explained in more detail in the next section.

3.5 3D Pose Fusion

The next goal is to determine the points in 3D space (world coordinates) given its projections onto the multiple images from the different camera perspectives. By knowing the camera matrix of the camera projection from world coordinates to screen coordinates, we know that each point in the image corresponds to a line in 3D space and every point on this line get projected on the same point in the image [35].

3.5.1 Triangulation

The method to solve this problem is called triangulation. In theory with perfect measurements this method is trivial. So let's assume we got a pair y_1 and y_2 of corresponding points in two different images. Remember that each point in the image belongs to a line in 3D space which are the green lines in the figure 3.8 [35]. The lines intersect at x which is the corresponding point in 3D space.

But in practice, the image points are measured with arbitrary accuracy. This leads to the two points y_1' and y_2' in the figure 3.8. If you consider the corresponding lines of these two points in 3D space (blue), they may not even intersect, and if they do, it is not correctly the corresponding 3D point x. Reasons for the divergence may be noisy measurements or an inaccurate

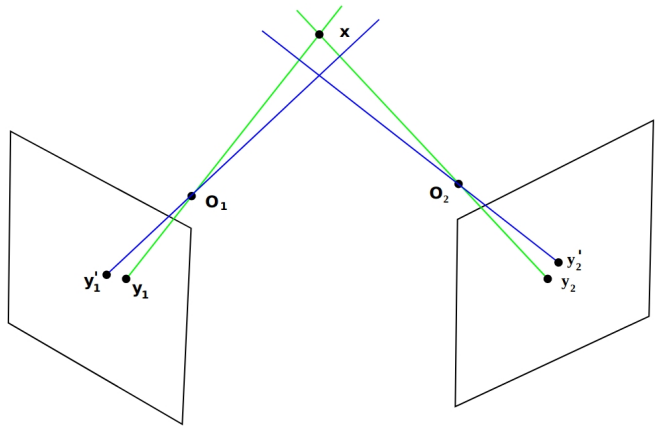


Figure 3.8: Triangulation visualization of two images with the camera's focal length O_1 and O_2. The ideal case with perfect measurements in green and the case of measurements with arbitrary accuracy in blue.
Source: [35]

camera calibration which lead to wrong focal lengths O_1 and O_2 and therefore to a wrong projection [35].

To find out which 3D point x' is the best estimate for the noisy measurements y_1' and y_2' you usually define an error measurement depending on x' and then minimize this error. Due to the fact that we do not always have all poses in 2D for the triangulation from the different views, and because the camera calibration is not error-free, we tried to use the Kalman Filter to do the triangulation. By constantly getting new measurements over time and the possibility of predicting the state further in time it seems to be a suitable application to estimate the 3D poses.

3.5.2 Extended Kalman Filter

The Kalman Filter (KF) is an iterative algorithm, based on the Bayesian Filter, that uses a series of measurements observed over time to find a good state estimation. The filter estimates the state using a form of feedback control where the measured values come into play. The theory in this section is based on the books *An Introduction to the Kalman Filter* [32] and *Kalman and Bayesian Filters in Python* [15].

The algorithm works in a two-step process, as shown in figure 3.9, which is repeated for every time step, or more precisely, for every frame. In the prediction step, the Kalman Filter projects the current state variable forward in time using the process model and projects the error covariance ahead. In the second step, called the update step, the previously estimated state

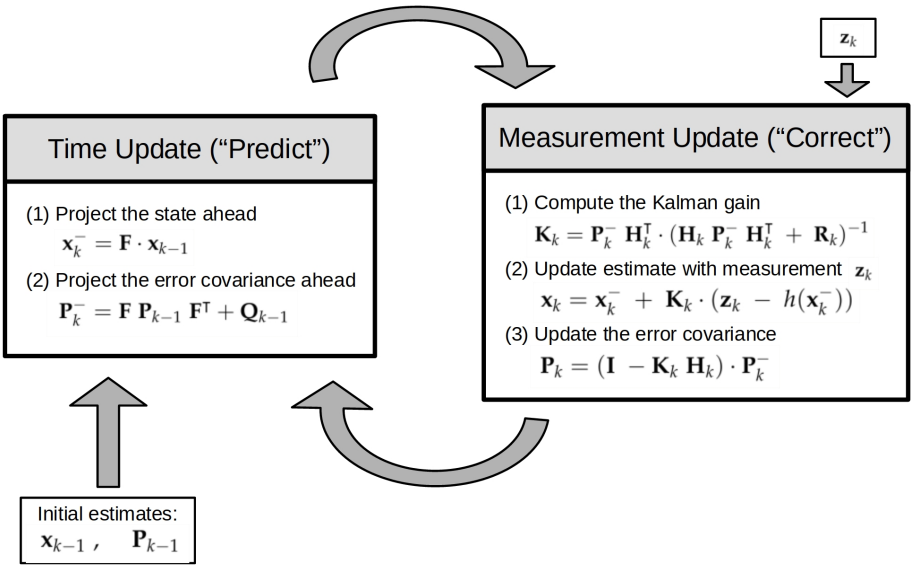


Figure 3.9: Complete picture of the calculations during the two-step process of the Extended Kalman Filter. For the Initialization the values of the first state vector and the error covariance are needed. A new measurement vector is given to the filter for each Measurement Update step. **Source:** Notation slightly adapted from [32]

variable gets updated using the measurement residual weighted with the so called Kalman gain. Additionally, the error covariance is updated.

In order to update the state variable, the residual is needed which is calculated from the current state variable and the newly added measurement $\mathbf{z}_k$. But the state variable is not part of the measurement space, therefore we have to project the state variable to the measurement space. This projection is basically a camera projection and is not linear due to a division, which is the reason why we use the Extended Kalman Filter (EKF). The reason why we work in measurement space and not in state space is because the measurements are not simply invertible which is exactly the goal of the EKF, to fuse the 2D measurements into the 3D space.

3.5.3 Designing the EKF

In the figure 3.9 and in the following text, the notation from the book *An Introduction to the Kalman Filter* [32] slightly adapted to the filterpy [15] notation will be used. We will omit the time subscript k for the definitions of multiple variables in this section for the sake of clarity. For coordinates x, y, z the subscript w is used for the 3D space and c_k for the screen space of the k-th camera. The state and measurement variable $\mathbf{x}, \mathbf{z}$ are written in bold to distinguish from the x and z coordinates. The last note is about the camera matrices which change with every frame. Thus, whenever the camera

calibration is mentioned, one always refers to the current calibration.

State and Measurement

The EKF is designed to track one person using multiple cameras. The number of cameras is defined in the initialization. As mentioned in section 3.3, the OpenPose model *COCO* consists of 18 keypoints. For the i-th keypoint the corresponding part of the state vector includes the position in 3D space (world coordinates) and their velocities:

$$\mathbf{x}^i := \begin{bmatrix} x_w^i & y_w^i & z_w^i & v_x^i & v_y^i & v_z^i \end{bmatrix}^\mathsf{T}, \quad \forall i \in \{0, 1, ..., 17\} \tag{3.6}$$

All 18 keypoints together form the state vector:

$$\mathbf{x} := \begin{bmatrix} \mathbf{x}^0 & \mathbf{x}^1 & \cdots & \mathbf{x}^{17} \end{bmatrix}^\mathsf{T} \tag{3.7}$$

The measurement vector $\mathbf{z}$ includes for the i-th keypoint the screen coordinates x, y for all n cameras:

$$\mathbf{z}^i := \begin{bmatrix} x_{c_0}^i & y_{c_0}^i & x_{c_1}^i & y_{c_1}^i & \cdots & x_{c_n}^i & y_{c_n}^i \end{bmatrix}^\mathsf{T}, \quad \forall i \in \{0, 1, ..., 17\} \tag{3.8}$$

The complete measurement vector for all 18 keypoints:

$$\mathbf{z} := \begin{bmatrix} \mathbf{z}^0 & \mathbf{z}^1 & \cdots & \mathbf{z}^{17} \end{bmatrix}^\mathsf{T} \tag{3.9}$$

Process Model

For the state propagation $\mathbf{F}$ in the Predict step we use a simple linear model that does not change over time. For the sake of simplicity we assume constant velocity. For several reasons this is not the best choice for modeling the real world. The players move in different directions, the head keypoints behave completely different compared to the hand keypoints etc. But due to the high frame rate, one can neglect these aspects because the movement from one frame to the next is minimal. The linear motion model for the x coordinates in 3D space is as follows $x = x + v_x \cdot \Delta t$. The time step Δt represents the time between two frames. The resulting state transition model F^i propagates the state vector for the i-th keypoint x^i further in time,

$$\mathbf{F}^i := \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \forall i \in \{0, ..., 17\} \tag{3.10}$$

so the resulting complete state propagation matrix $\mathbf{F}$ is a diagional matrix expressed using multiple submatrices $\mathbf{F}^i$:

$$\mathbf{F} := \begin{bmatrix} \mathbf{F}^1 & & \\ & \ddots & \\ & & \mathbf{F}^{17} \end{bmatrix} \tag{3.11}$$

This results in the first equation of the extended Kalman filter to project the state ahead:

$$\mathbf{x}_k^- = \mathbf{F} \cdot \mathbf{x}_{k-1} \tag{3.12}$$

Together with the next equation 3.13, those two form the prediction step of the EKF. The estimation error covariance $\mathbf{P}$ gets projected from the previous time step $k-1$ to the current time step k.

$$\mathbf{P}_k^- = \mathbf{F}\ \mathbf{P}_{k-1}\ \mathbf{F}^\mathsf{T} + \mathbf{Q}_{k-1} \tag{3.13}$$

At the beginning the covariance matrix $\mathbf{P}$ has to be initialized and afterwards the filter takes care of updating its value. We know that the position and velocity are correlated but due to the fact that we use constant velocity in our model we just initialize $\mathbf{P}$ to zero. The matrix is a square matrix of the same dimension as the state vector.

Noise

In both equation 3.13 and 3.15 we add a covariance matrix to the value derived from white Gaussian noise. $\mathbf{Q}$ is a square matrix of the same dimension as the state vector $\mathbf{x}_k$ and is called the process noise covariance. The matrix $\mathbf{R}$ is also a square matrix with the dimension of the measurement vector $\mathbf{z}_k$ and is called the measurement noise covariance. They're called white Gaussian noise because they are independent of each other with normal probability distribution. During the EKF iteration they stay constant with values $\sigma_q = 0.5$ and $\sigma_r = 0.1$ which were obtained by trying out different values:

$$\mathbf{Q} := \begin{bmatrix} \sigma_q^2 & & \\ & \ddots & \\ & & \sigma_q^2 \end{bmatrix} \qquad \mathbf{R} := \begin{bmatrix} \sigma_r^2 & & \\ & \ddots & \\ & & \sigma_r^2 \end{bmatrix} \tag{3.14}$$

The reason for the higher process noise is the inaccurate process model with constant velocity. So the filter tries to rely more on the measurements than on the prediction as desired.

Kalman gain and Error covariance

The residual gets added to the predicted state variable $\mathbf{x}_k^-$ as described in the next section. But the residual is weighted by the Kalman gain. Higher values for the Kalman gain indicate that we give more trust to the measurement and lower values indicate that we rely more on the prediction. So if the measurement uncertainty $\mathbf{R}_k$ is smaller with respect to the error covariance $\mathbf{P}_k^-$ we rely more on the measurement. The computation for the Kalman gain takes place in the Measurement Update step:

$$\mathbf{K}_k = \mathbf{P}_k^- \ \mathbf{H}_k^\intercal \cdot (\mathbf{H}_k \ \mathbf{P}_k^- \ \mathbf{H}_k^\intercal \ + \ \mathbf{R}_k)^{-1} \tag{3.15}$$

Another equation in the Measurement Update step is the update of the error covariance $\mathbf{P}_k$.

$$\mathbf{P}_k = (\mathbf{I} \ - \mathbf{K}_k \ \mathbf{H}_k) \cdot \mathbf{P}_k^- \tag{3.16}$$

In contrast to $\mathbf{Q}$ and $\mathbf{R}$ which are both constant, the error covariance $\mathbf{P}_k$ and the Kalman gain $\mathbf{K}_k$ will stabilize quickly and then remain constant as described in [32]. These two equations involve the matrix $\mathbf{H}$ which is used for the linearization of the function h. This is explained in more detail in section 3.5.3.

Measurement Model

The measurement model is designed through a function h which is responsible to relate the state vector $\mathbf{x}_k$ to the measurement $\mathbf{z}_k$. As we mentioned in section 3.5.2 this function projects the state variable into the measurement space in order to be able to compute the residual which leads to this equation for the Measurement Update step:

$$\mathbf{x}_k = \mathbf{x}_k^- \ + \ \mathbf{K}_k \cdot (\mathbf{z}_k \ - \ h(\mathbf{x}_k^-)) \tag{3.17}$$

The explanation for what the function h looks like and what the related matrix $\mathbf{H}_x$ is will follow in detail. The filterpy implementation [16] of the EKF requires for the update step the function h with the arguments $\mathbf{x}_k$ and the camera matrices for all cameras. The output of this function is a vector which looks exactly like the measurement vector $\mathbf{z}_k$, but it is just the same space and no new measurement vector:

$$h(\mathbf{x}_k^-, \text{camera matrices}) = \mathbf{z}_{transformed} \tag{3.18}$$

As mentioned in section 3.1, a homogeneous point in the 3D space can be projected to the image screen using the camera matrix $\mathbf{M}$. We will explain this step for the x_w, y_w, z_w coordinates of the state vector $\mathbf{x}_k$ for one keypoint to map to the corresponding measurement space of one specific camera j.

We will omit the time subscript k in this section for the sake of understanding.

$$\mathbf{M}_{c_j} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \mathbf{A}_{c_j} \left[\begin{array}{c|c} \mathbf{R}_{c_j} & \mathbf{T}_{c_j} \end{array} \right] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3.19}$$

$$= \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3.20}$$

$$= \begin{bmatrix} x'_{c_j} \\ y'_{c_j} \\ z'_{c_j} \end{bmatrix} \tag{3.21}$$

In order to get the correct scale and dimension for the screen coordinates x_{c_j}, y_{c_j} one has to divide the result from equation 3.21 by z_{c_j} and drop the last entry for the z coordinate. This forms a part of the function h:

$$\hat{h}(\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}, \mathbf{A}_{c_j}, \mathbf{R}_{c_j}, \mathbf{T}_{c_j}) = \begin{bmatrix} x_{c_j} \\ y_{c_j} \end{bmatrix} \tag{3.22}$$

$$= \begin{bmatrix} \frac{x'_{c_j}}{z'_{c_j}} \\ \frac{y'_{c_j}}{z'_{c_j}} \end{bmatrix} = \begin{bmatrix} \frac{f_x \cdot (r_{1,1} \cdot x_w + r_{1,2} \cdot y_w + r_{1,3} \cdot z_w + t_x)}{r_{3,1} \cdot x_w + r_{3,2} \cdot y_w + r_{3,3} \cdot z_w + t_z} + x_0 \\ \frac{f_y \cdot (r_{2,1} \cdot x_w + r_{2,2} \cdot y_w + r_{2,3} \cdot z_w + t_x)}{r_{3,1} \cdot x_w + r_{3,2} \cdot y_w + r_{3,3} \cdot z_w + t_z} + y_0 \end{bmatrix} \tag{3.23}$$

To make it clear once again, this was just to transform the 3D coordinates for one keypoint into the image space of one camera. So for the i-th keypoint of the state vector $\mathbf{x}_k$ we drop the velocities and transform the 3D coordinates to screen coordinates for each of the n cameras. This gives us a bigger part of h:

$$h_i(\begin{bmatrix} x_w^i \\ y_w^i \\ z_w^i \end{bmatrix}, \text{ camera matrices}) = \begin{bmatrix} \hat{h}(\begin{bmatrix} x_w^i \\ y_w^i \\ z_w^i \end{bmatrix}, \mathbf{A}_{c_0}, \mathbf{R}_{c_0}, \mathbf{T}_{c_0}) \\ \vdots \\ \hat{h}(\begin{bmatrix} x_w^i \\ y_w^i \\ z_w^i \end{bmatrix}, \mathbf{A}_{c_n}, \mathbf{R}_{c_n}, \mathbf{T}_{c_n}) \end{bmatrix} \tag{3.24}$$

Finally we can formulate the whole function h for all 18 keypoints:

$$h(\mathbf{x}_k^-, \text{camera matrices}) = \begin{bmatrix} h_0(\begin{bmatrix} x_w^0 \\ y_w^0 \\ z_w^0 \end{bmatrix}, \text{ camera matrices}) \\ \vdots \\ h_{17}(\begin{bmatrix} x_w^{17} \\ y_w^{17} \\ z_w^{17} \end{bmatrix}, \text{ camera matrices}) \end{bmatrix} \tag{3.25}$$

$$= \mathbf{z}_{transformed} \tag{3.26}$$

Jacobian Matrix H

Because of the fact that the relationship between state space and the measurement space is a nonlinear function h, we need to linearize it by evaluating its partial derivatives with respect to the state vector $\mathbf{x}_k$ as a Taylor series. The resulting matrix is the Jacobian matrix $\mathbf{H}_k$. What the derivations of one screen coordinate pair (the i-th keypoint) like in equation 3.22 for the camera j looks like is explained as follows:

$$\mathbf{H}_{c_j}^i = \begin{bmatrix} \frac{\partial x_{c_j}^i}{\partial x_w^i} & \frac{\partial x_{c_j}^i}{\partial y_w^i} & \frac{\partial x_{c_j}^i}{\partial z_w^i} & \frac{\partial x_{c_j}^i}{\partial v_x^i} & \frac{\partial x_{c_j}^i}{\partial v_y^i} & \frac{\partial x_{c_j}^i}{\partial v_z^i} \\ \frac{\partial y_{c_j}^i}{\partial x_w^i} & \frac{\partial y_{c_j}^i}{\partial y_w^i} & \frac{\partial y_{c_j}^i}{\partial z_w^i} & \frac{\partial y_{c_j}^i}{\partial v_x^i} & \frac{\partial y_{c_j}^i}{\partial v_y^i} & \frac{\partial y_{c_j}^i}{\partial v_z^i} \end{bmatrix} \tag{3.27}$$

The matrix $\mathbf{H}_{c_j}^i$ would be much wider because of the partial derivatives for the other state variables but they are all zero, so we just note this block matrix and build the Jacobian $\mathbf{H}$ with multiple similar blocks. So the whole corresponding block of the Jacobian for the i-th keypoint with respect to all n cameras is presented:

$$\mathbf{H}^i = \begin{bmatrix} \mathbf{H}_{c_0}^i \\ \mathbf{H}_{c_1}^i \\ \vdots \\ \mathbf{H}_{c_n}^i \end{bmatrix} \tag{3.28}$$

And finally we can formulate the entire Jacobian $\mathbf{H}_k$ with the partial derivatives for the entire state vector, note that in the steps above the subscript k has been removed for readability.

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{H}^0 & & \\ & \ddots & \\ & & \mathbf{H}^{17} \end{bmatrix} \tag{3.29}$$

Like the function h the filterpy [16] implementation also requires the Jacobian of h as input for the update step. So the Jacobian $\mathbf{H}_x$ is also a function

with the same arguments as the function h, namely the state vector $\mathbf{x}_k$ and the camera matrices. For the readability we first define some variables which are used afterwards:

$$r_1\mathbf{x}^i := r_{1,1} \cdot x_w^i + r_{1,2} \cdot y_w^i + r_{1,3} \cdot z_w^i \tag{3.30}$$

$$r_2\mathbf{x}^i := r_{2,1} \cdot x_w^i + r_{2,2} \cdot y_w^i + r_{2,3} \cdot z_w^i \tag{3.31}$$

$$r_3\mathbf{x}^i := r_{3,1} \cdot x_w^i + r_{3,2} \cdot y_w^i + r_{3,3} \cdot z_w^i \tag{3.32}$$

So what's left are the partial derivatives. The computation for those from equation 3.27 are shown with respect to the i-th keypoint and the j-th camera, the others are analog but with the corresponding keypoint coordinates and the corresponding camera matrices.

$$\frac{\partial x_{c_j}^i}{\partial x_w^i} = \frac{f_x \cdot r_{1,1}}{r_3\mathbf{x}^i + t_z} - \frac{f_x \cdot r_{3,1}(r_1\mathbf{x}^i + t_x)}{(r_3\mathbf{x} + t_z)^2} \tag{3.33}$$

$$\frac{\partial x_{c_j}^i}{\partial y_w^i} = \frac{f_x \cdot r_{1,2}}{r_3\mathbf{x}^i + t_z} - \frac{f_x \cdot r_{3,2}(r_1\mathbf{x}^i + t_x)}{(r_3\mathbf{x} + t_z)^2} \tag{3.34}$$

$$\frac{\partial x_{c_j}^i}{\partial z_w^i} = \frac{f_x \cdot r_{1,3}}{r_3\mathbf{x}^i + t_z} - \frac{f_x \cdot r_{3,3}(r_1\mathbf{x}^i + t_x)}{(r_3\mathbf{x} + t_z)^2} \tag{3.35}$$

$$\frac{\partial y_{c_j}^i}{\partial x_w^i} = \frac{f_y \cdot r_{2,1}}{r_3\mathbf{x}^i + t_z} - \frac{f_y \cdot r_{3,1}(r_2\mathbf{x}^i + t_y)}{(r_3\mathbf{x} + t_z)^2} \tag{3.36}$$

$$\frac{\partial y_{c_j}^i}{\partial y_w^i} = \frac{f_y \cdot r_{2,2}}{r_3\mathbf{x}^i + t_z} - \frac{f_y \cdot r_{3,2}(r_2\mathbf{x}^i + t_y)}{(r_3\mathbf{x} + t_z)^2} \tag{3.37}$$

$$\frac{\partial y_{c_j}^i}{\partial z_w^i} = \frac{f_y \cdot r_{2,3}}{r_3\mathbf{x}^i + t_z} - \frac{f_y \cdot r_{3,3}(r_2\mathbf{x}^i + t_y)}{(r_3\mathbf{x} + t_z)^2} \tag{3.38}$$

$$\frac{\partial x_{c_j}^i}{\partial v_x^i} = \frac{\partial x_{c_j}^i}{\partial v_y^i} = \frac{\partial x_{c_j}^i}{\partial v_z^i} = \frac{\partial y_{c_j}^i}{\partial v_x^i} = \frac{\partial y_{c_j}^i}{\partial v_y^i} = \frac{\partial y_{c_j}^i}{\partial v_z^i} = 0 \tag{3.39}$$

State Initialization

For the state initialization we need three dimensional locations of the poses. In order to find a satisfying initial guess, we take the uniform 3D skeletons based on the x, y location on the pitch for the first time step from the 2D tracking data. For the velocities we set $v_x = v_z = 3.0$ and for the y-axis perpendicular to the pith $v_y = 0.0$ for every keypoint.

Summary

All equations together result in the final EKF. A overview for the five equations of the filter are repeated here:

The Update ("Predict"):

(1) Projet the state ahead:

$$\mathbf{x}_k^- = \mathbf{F} \cdot \mathbf{x}_{k-1} \tag{3.40}$$

(2) Project the error covariance ahead:

$$\mathbf{P}_k^- = \mathbf{F}\ \mathbf{P}_{k-1}\ \mathbf{F}^\intercal + \mathbf{Q}_{k-1} \tag{3.41}$$

Measurement Update ("Correct"):

(1) Compute the Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_k^-\ \mathbf{H}_k^\intercal \cdot (\mathbf{H}_k\ \mathbf{P}_k^-\ \mathbf{H}_k^\intercal\ +\ \mathbf{R}_k)^{-1} \tag{3.42}$$

(2) Update estimate with measurement $\mathbf{z}_k$:

$$\mathbf{x}_k = \mathbf{x}_k^-\ +\ \mathbf{K}_k \cdot (\mathbf{z}_k\ -\ h(\mathbf{x}_k^-)) \tag{3.43}$$

(3) Update the error covariance:

$$\mathbf{P}_k = (\mathbf{I}\ -\mathbf{K}_k\ \mathbf{H}_k) \cdot \mathbf{P}_k^- \tag{3.44}$$

These equations of the Extended Kalman Filter fuse the human 2D poses of different camera perspectives into the corresponding 3D poses in world coordinates. The results are presented and discussed in the next chapter.

3.6 Problems

The last section about the Extended Kalman Filter was written with regard to perfect measurements $\mathbf{z}_k$, but we have seen in section 3.4 that this is not the case. The resulting 2D poses which form the measurement vectors are not complete because some poses lack keypoints or the whole pose is missing. To address this problem we have to check the structured 2D poses before passing them as new measurement to the Extended Kalman Filter. So we check the 2D poses from *OpenPose* in the following manner:

- If the keypoint is not available at all, we use the three-dimensional prediction of the EKF and project them to screen coordinates.
- Otherwise, we test every single keypoint if the confidence is equal to zero.

For the second case we try to estimate some missing keypoints proportional to the given keypoints, for example when one shoulder or some head keypoints are missing. For the others we take the prediction of the keypoint as described in the first case above.

The neck keypoint is always given because the refinement step requires this. Therefore we can estimate the missing shoulder keypoint proportional to the neck and the other shoulder keypoint assuming they are placed in a straight line. In the same fashion we estimate the missing keypoints of the head. For this we assume the head is positioned normal and not leaning back or forward. With the help of the other existing keypoints, we estimate the two-dimensional screen coordinates proportional of the given ones. This heuristic allows us to avoid using the predictions as new measurements for the missing keypoints.

4 Results and Discussion

This chapter presents the final results and the discussion of the individual intermediate steps with the respective results to show existing limitations. All the results are based on the video material provided by the Swiss national soccer team and their 2D tracking data.

3D Pose Estimation

The estimated three-dimensional pose estimations are projected to screen coordinates in order to display them on the images. As mentioned in section 3.1, the camera calibration was slightly improved for the camera K8 and K9. For this reason we present here the results of the Extended Kalman Filter using the data obtained by those two cameras. The design of our Kalman Filter allows for a variable amount of cameras, which we may specify during initialization.

Figure 4.1: Projection of the estimated three-dimensional poses into the screen space for camera K8.

In figure 4.1 the projection of the estimated three-dimensional poses into screen space for camera K8 are shown for one of the first frames. This already shows important limitations, the players in the upper left corner are out of place. Due to the fact that we do not have two-dimensional pose estimations for the players in every frame, especially for the players far away,

we concentrate on the results of the players where the poses are mostly recognized, such as the players in the middle of the pitch.

For the discussion below we will focus on the player marked in figure 4.2 because the estimated two-dimensional keypoints are almost always available.

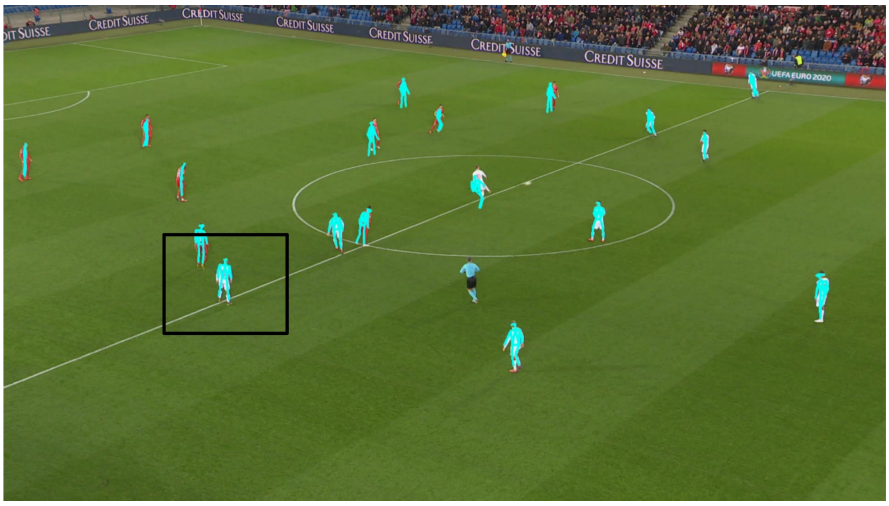


Figure 4.2: Projection of the estimated three-dimensional poses into the screen space for camera K9.

The projection of the three-dimensional pose for this player are shown in figure 4.3 and 4.4 for the two cameras. The captured scene is an example for a situation where the estimation works well. In general, the algorithm

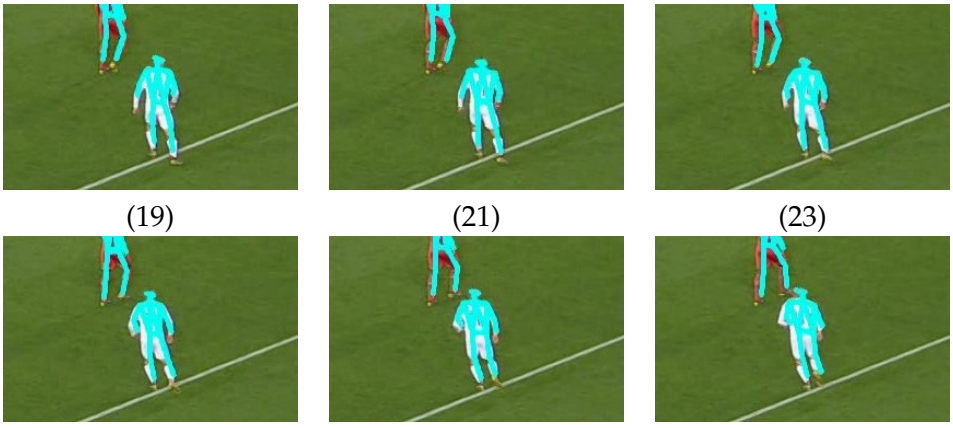


Figure 4.3: Projection of the three-dimensional keypoints of the selected player into screen space for camera K9

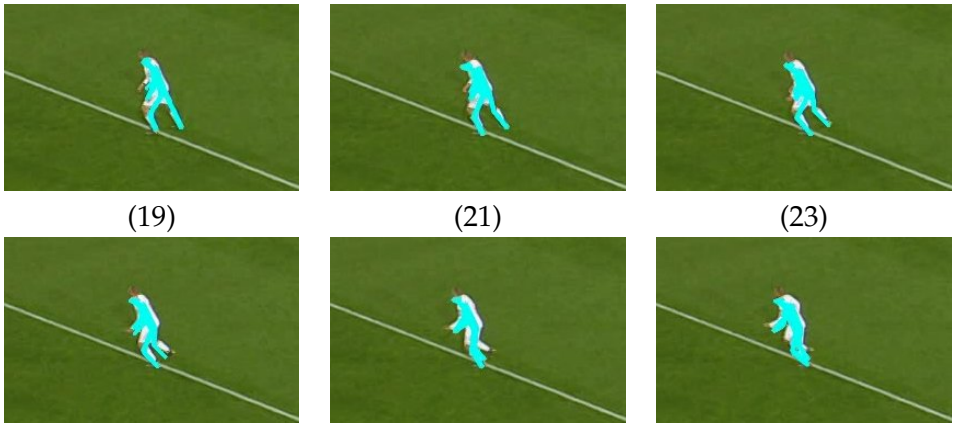


Figure 4.4: Projection of the same player as in figure 4.3 but into screen space for camera K8.

requires many improvements in order to work for longer scenes and every player on the pitch. Below we present multiple situations where the Extended Kalman Filter estimates wrong three-dimensional poses and discuss the possible reasons for it.

In figure 4.5 we have a scene where one body is mostly concealed by another player. In frame (06) *OpenPose* recognized the pose of the player in the front and therefore the filter has a true measurement. The player in the background has still no two-dimensional pose estimations and therefore the projected uniform skeleton is used as measurement which moves right due to the process model. This described scenario is repeated for the next few

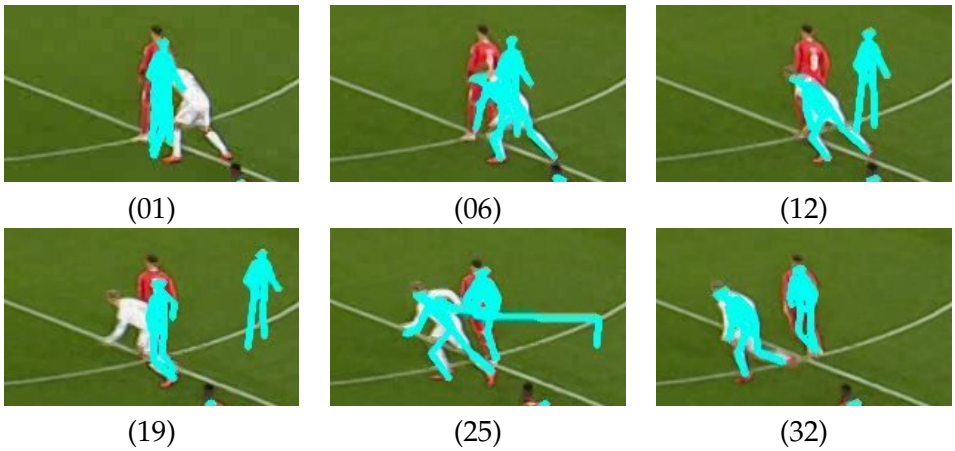
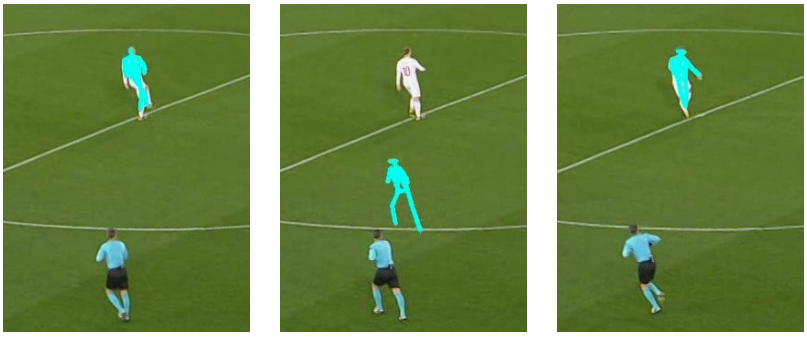


Figure 4.5: Two players close together where some body parts are concealed by others. The images show the projection of the estimated three-dimensional poses for the two players to screen space for the camera K8.

frames up to and including (12), afterwards *OpenPose* recognize a mixture out of the two players as new measurement and the same skeleton as before (19). Finally in frame (25) the filter receives measurements for both players. But there is still a leg on the right side of the red player due to the prediction of the skeleton since the measurement pose did not contain the keypoints for the left knee and foot. Later in frame (32) no body part is concealed and therefore the complete three-dimensional pose can be estimated. The above situation is one example where the keypoints delivered by *OpenPose* are insufficient for estimating correct three-dimensional poses.

As already mentioned we are not able to estimate the 3D poses for the players in the left upper corner due to missing two-dimensional measurements. Another similar problem are poses for the players which were not covered by both cameras such as the left most players in figure 4.2. The filter requires at least two different perspectives to estimate the three-dimensional poses, otherwise the filter is not able to estimate the lost depth.

Another problem is the mapping of the measurements to the correct player. In figure 4.6 the three-dimensional pose is correctly estimated for frame (16) but for frame (23) *OpenPose* did not recognize a pose for the player, so we assign the pose with the smallest distance, which is in this case the referee's pose. In frame (23) the projected 3D pose is inbetween the referee and the player. This happens because the EKF is combining the last estimation, which was correct, with the new measurement (the referees pose). As soon as the filter receives a correct measurement, the estimated three-dimensional pose is correct again as in frame (35). This problem could be mitigated by testing whether the new measurements belong to the correct player by considering the distance between consecutive measurements.



(16) (23) (35)

Figure 4.6: Referee's pose is used as new measurement while *OpenPose* did not recognized the pose for the player at the given frame.

For some reason after a given time the results start to drift away as demonstrated in figure 4.7. At the beginning the poses fit well and then the pro-

jected three-dimensional poses start to move to the right while the pose itself is correct but the location in world coordinates is somehow misplaced. We think this issue might be related to the other issues described above, but further investigation is required in this particular case.

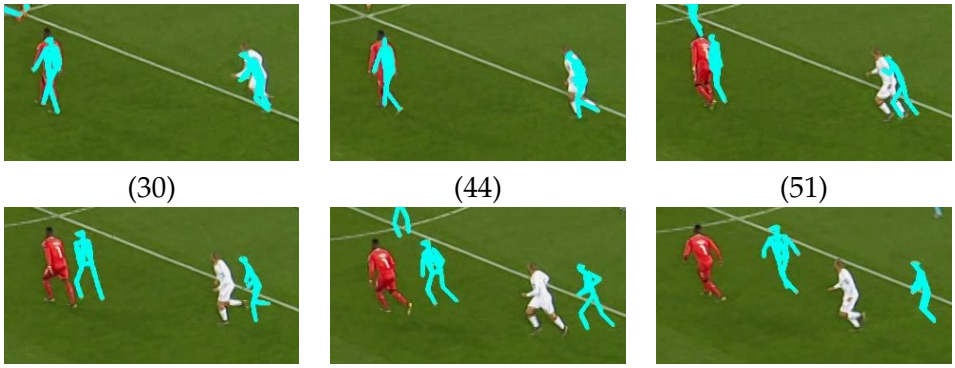


Figure 4.7: After a given time the estimated three-dimensional poses drift to the right.

The next results in figure 4.8 are used to discuss the differences between the filter results for two cameras compared to the results with the additional central camera K1. One would expect an additional camera to be an advan-

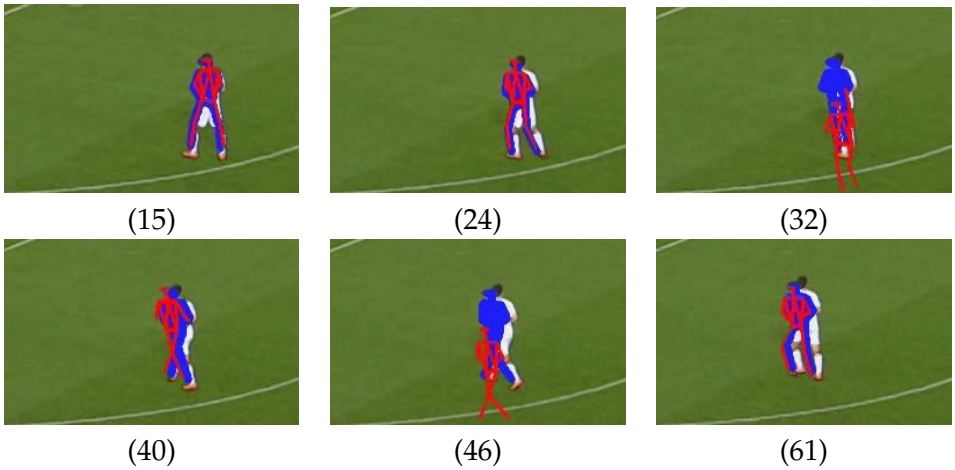
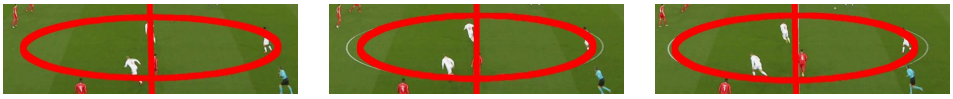


Figure 4.8: Projection of the estimated three-dimensional poses, the blue poses were estimated with the use of camera K8 and K9 while the red poses had the additional data from the central camera K1.

tage to obtain more measurements for the filter, resulting in more robust estimated poses. But as i mentioned we could not use the improved calibration method for the additional central camera K1. The impact of a slightly

disturbed camera calibration is directly shown in figure 4.8. The blue poses correspond to the estimations of the filter for two cameras and the red ones to the estimations for three cameras. The red pose is slightly off in frame (32) and (46) due to a trembling calibration for those frames, which can be seen in the calibration in figure 4.9.



(24) (32) (46)

Figure 4.9: Calibration for the central camera K1 for the frames where the poses are slightly off.

Calibration

An important point for the whole process is a correct camera calibration for each frame, in contrast to other projects the cameras are not explicitly installed for this purpose. The provided video material is recorded with TV cameras, these pan around and zoom in and out. Therefore the camera matrices obtained by the calibration belong to the specific frame and are different for each frame. Static cameras, on the other hand, are calibrated beforehand or with the use of the first frame and remain constant later.

In figure 4.10 a few frames out of the first 25 are shown for the camera K1. In order to show the calibration the soccer field is projected onto the image. The figure demonstrate the working calibration but even for the first few frames where the camera stays mostly static you can see how the soccer field trembles a little for example in frame 014 or even more in frame 024. The situation got worse in later scenes when the camera starts to zoom and pan. As mentioned in section 3.1 we used an improved calibration method for the cameras K8 and K9. This prevented the calibration from small trembles as the given example from image (024). So for the three cameras presented in the method chapter the calibration remains fairly stable for the first few seconds but the important point is that the calibration always recover after a bad calibration. But this works just for the first few seconds, afterwards the calibration is lost and can not recover. This behavior is shown for for camera K9 in figure 4.11. The camera pans to the left side and zooms in at the same time for which the algorithm cannot properly propagate the calibration from frame to frame and converges to an incorrect calibration from which it can no longer recover. These results limit our approach to process longer sequences than just the first few seconds. But nevertheless we continued with the given calibration for the first few seconds with the next steps.

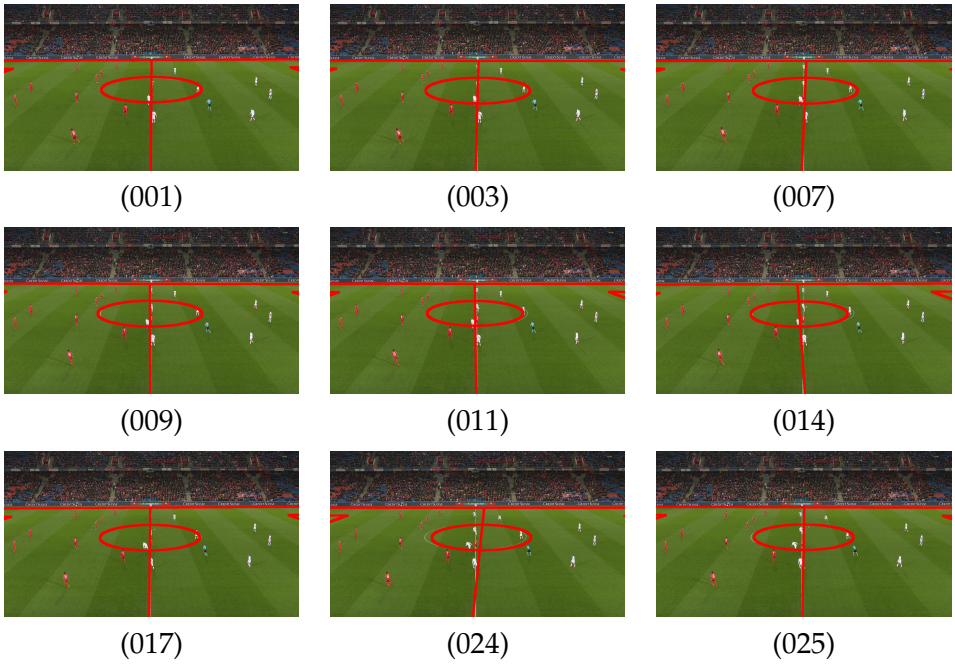


Figure 4.10: Projection of the soccer field onto the frames with the corresponding camera calibration from camera K1.

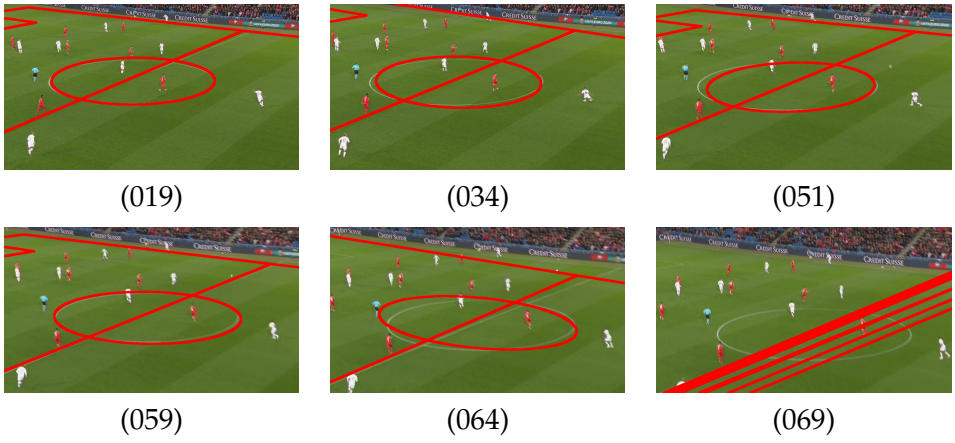


Figure 4.11: Projection of the soccer field onto the frames with the corresponding camera calibration from camera K9 to demonstrate the failing calibration.

Early on we mentioned that we have also access to other cameras such as the GoPros and other cameras provided by the Swiss national soccer team. Unfortunately most of the cameras are zoom cameras which are terrible to calibrate but could be useful in the future for an improved system for specific scenes in order to boost the accuracy. Two other cameras looked quite promising, a left and right overview camera. But just like the GoPros

they were equipped with fish eye lenses and the calibration tries to map a pinhole camera model to it as seen in figure 4.12. Because the cameras are static and offer a good overview the system would benefit from those views. But without a proper calibration we were not able to further use them in later steps.

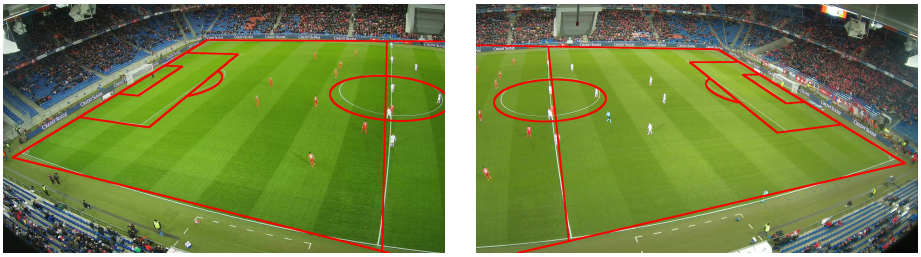


Figure 4.12: Calibration of the two overview cameras.

Player Detection

Detectron delivers the detected objects as displayed in figure 4.13. The algorithm works almost perfectly and detects for the most time all players on the pitch. As mentioned in 3.2 *Soccer on your Tabletop* modified the used subroutine from figure 4.13 slightly in order to extract the segmentation mask and the bounding boxes. Sometimes the algorithm detects not all players which

Figure 4.13: Result from the original subroutine provided by *Detectron*[10]

is not a big problem by the fact that the used crops for the pose estimation are quite big and therefore the undetected players are most of the time still in a crop from another player. Thus small errors in this step are negligible.

In figure 4.14 a few special cases are shown. The case (1) is a good example for the described situation above and is therefore not a problem, the two other cases (2) and (3) are also not problematic because of the described method in section 3.4 to remove poses for the same person.

(1) not detected (2) multiple boxes (3) overlapping

Figure 4.14: Bounding boxes examples for a few situations.

The only problem are scenes as shown in figure 4.15 where no bounding boxes are detected. This results in missing crops for the 2D pose estimation. However, the situation is not so bad because the sequences with missing boxes are only for a few frames or mostly only for one frame.

Figure 4.15: Example for consecutive frames where the bounding boxes are not always recognized.

2D Pose Estimation

In addition to the camera calibration, this step lays the foundations for the Extended Kalman Filter. Without proper two-dimensional pose estimations the filter has no new measurements and must rely on the prediction. The results from *OpenPose* vary greatly from frame to frame. In figure 4.16 the estimated poses for a few cutouts for the first frame from the camera K1 are shown without the refinement step and in figure 4.17 the complete scene after the poses were refined. This illustrates nicely the process of the 2D refinement, complete estimated players are taken over as for example (1) or (5), while the player top right in the cutout (2) is also estimated in the center (4) and the performed neck non-maximal-suppression algorithm discarded the part estimation from (2) and saved the one from (4). Another good example is the right player in (3) which seems to be recognized because of the drawn connections of the keypoints but there were not enough keypoints

to keep the player for the refined version (the player on the far right).

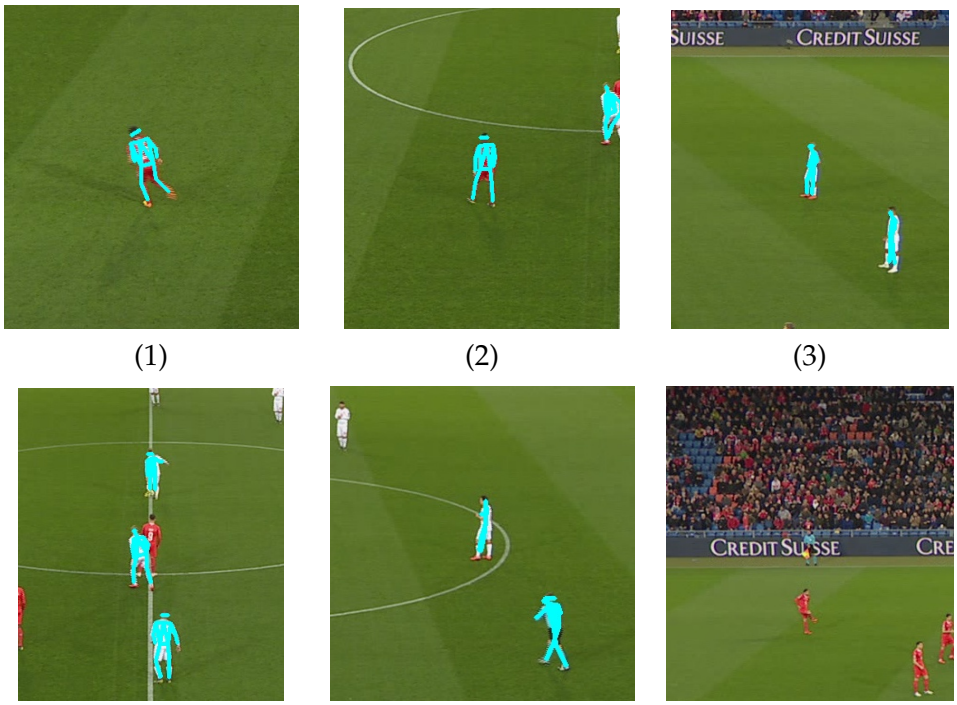


Figure 4.16: *OpenPose* results on the cutouts from the bounding boxes for camera K1.

Figure 4.17: *OpenPose* results from the cutouts in the full scene after the pose refinement step for camera K1.

Unfortunately *OpenPose* is not able to estimate the poses for the players far away as the ones in cutout (6) or for players who are partially covered by another player as the one in cutout (4). Another problem are the head keypoints (14-17) of the COCO pose format, these were rarely all recognized. A solution to solve this problem to some extent was presented in section 3.6, which seems more like a workaround.

Figure 4.18: Refined poses for the first frame from camera K8.

Figure 4.19: Refined poses for the first frame from camera K9.

This leads to a further limitation of the algorithm with the given data and their camera perspectives. In order to properly observe all the keypoints of the players the use of additional cameras, with the same perspectives

as the three cameras used but from the other side of the pitch, would be very useful. This would solve the problem to estimate the 2D poses for the players far away on the pitch where we have no continuous measurements. To achieve this we set up two GoPro cameras in Basel on the other side of the pitch. But the quality of these cameras for the long distance to the soccer field was not good enough and also the calibration for fish eye lenses is not yet solved.

As described in section 3.4 we used the tracking data and uniform skeletons to map the detected poses to a specific player on the pitch. The projected 3D skeletons are shown in figure 4.20 for the same frame as the recognized poses in figure 4.19 for the camera K9. The skeleton's position does not always match with the location of the player on the pitch but by using them for the mapping they fulfill their purpose.

Figure 4.20: Projection of the uniform 3D skeletons into the image space for camera K9.

Setup and Runtime

Due to the fact that the algorithm consists of several steps which requires a lot of computational power we used the *Leonhard Cluster* for the object detection. Afterwards the calibration and 2D pose estimation were performed on a Nvidia Geforce GTX 970. The 3D pose estimation requires no dedicated graphics card and was therefore also performed on a DELL XPS 13 to test out different parameters. To provide an overview for the duration of the algorithm the time required is listed. The runtime for *Detectron* is about 3 seconds ,*OpenPose* about 5 seconds and the calibration about one second per frame. This results in rougly 9 seconds per frame and for three cameras 27 seconds. The 3D pose estimation requires less than a second per

iteration which is equivalent to a frame. In total, the algorithm processes a one-minute video with 25 fps in about 70 minutes. However, the runtime does not matter so far and therefore we have not explicitly tried to improve it.

In order to get the pipeline working pay attention to properly install the prerequisites of all building blocks. Additionally pay attention to the python dependencies. This step takes some time to get everything to work.

5 Conclusion

The goal of this work was to explore the state-of-the-art computer vision and visualization technology to analyze soccer games with the given data. The application of computer vision based technologies in sports is more and more present today. So a system is presented that estimates the three-dimensional human poses using different camera perspectives and two dimensional tracking data of a soccer game. The main building blocks of the systems are a object detection algorithm[10] to improve the results of the human 2D pose estimations[2]. For the calibration, provided by the *Soccer on your Tabletop* project, a manual initialization for the first frame is used and afterwards propagated using the previous calibration and a line detection algorithm. The final 3D pose fusion used an Extended Kalman Filter which predicts the next three-dimensional state of the poses and corrects its guess with the use of the constructed human 2D poses.

The results show the feasibility of setting up a system to estimate three-dimensional poses without large investments compared to other existing high-quality systems in this area. This work serves as a feasibility study and therefore we discovered many limitations such as the camera calibration, not enough camera angles to completely track what is happening on the pitch and a few more as described in the chapter 4. These results are of course not yet accurate enough to work with but nevertheless they showed that the goal is achievable with further improvements. By the fact that our calibration did not work very well we were very limited. Firstly the calibration did not work for scenes where the tv cameras pan fast or zoom strongly so we were just able to test our system with scenes where the camera remains reasonably stable. Secondly the calibration was even then not stable enough to provide a good basis for the Extended Kalman Filter. So in order to develop a system for the 3D pose estimation for the whole soccer game with a good accuracy there are a lot of points to further work on.

Future Work

For further research, the most important part would be a stable camera calibration. The best way to achieve this would be static cameras that only need to be calibrated once and not for every frame. The provided perspectives were good but as mentioned in chapter 4, additional static cameras on the other side of the pitch and the replacement of the TV cameras with static cameras would help a lot. This would require an investment, but would

be an easy way to solve the calibration problems. Otherwise, one could improve the camera calibration by trying to add the camera parameters to the EKF and smooth the results. If the camera pans and zooms without strong changes of direction, this would maybe work quite well. Another point to mention here is the problem to calibrate the fish eye cameras, if we were able to calibrate them we would have access to more measurements for the Extended Kalman Filter.

To improve the estimated two-dimensional keypoints, the additional cameras, as described above, would help a lot to recognize every player. But in order to recognize even the most distant players one could try to use smaller up sampled crops for better results.

In order to boost the Extended Kalman Filter results, a suitable motion model is required to predict the keypoints with a better precision, which is important for scenes where no keypoints are available and the filter has to rely only on the predictions. For this purpose, the constant velocity assumption has to be removed and replaced with an accurate process model.

At the point where the system works properly with a desired accuracy, one can use the generated three-dimensional poses to analyze the game in more detail. For example a player's point of view can be analyzed for certain situations or the system can even be used for automatic offside decisions because these are based on body parts, which is a perfect example.

Acknowledgments

I would like to thank my supervisor Dr. Martin Oswald for all the spontaneous meetings and prompt support in case of problems. I also acknowledge the effort from the authors of the state-of-the-art algorithms/libraries such as *Soccer on your Tabletop*, *OpenPose*, *Detectron* and *FilterPy* which may be used for research projects.

Bibliography

[1] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter V. Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: automatic estimation of 3d human pose and shape from a single image. *CoRR*, abs/1607.08128, 2016.

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1812.08008, 2018.

[3] cedro-blog. TensorFlow OpenPose. http://cedro3.com/ai/tensorflow-openpose/. [11-August-2019].

[4] OpenCV Documentation. Camera calibration and 3d reconstruction. https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

[5] OpenCV Documentation. Canny edge detection. https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html.

[6] OpenCV Documentation. Image segmentation with distance transform and watershed algorithm. https://docs.opencv.org/3.4.3/d2/dbd/tutorial_distance_transform.html.

[7] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2366–2374. Curran Associates, Inc., 2014.

[8] FIFA. 2018 World Cup Russia. https://www.fifa.com/worldcup/news/more-than-half-the-world-watched-record-breaking-2018-world-cup. [3-August-2019].

[9] Ross Girshick. Github: Non maximum suppression. https://github.com/rbgirshick/py-faster-rcnn/blob/master/lib/nms/cpu_nms.pyx.

[10] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018.

[11] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CoRR*, abs/1609.03677, 2016.

[12] Intel. True view. https://www.intel.co.uk/content/www/uk/en/sports/technology/true-view.html. [3-August-2019].

[13] Intel. Video: Arsenal fc adds intel true view technology. https://newsroom.intel.com/video-archive/video-arsenal-fc-adds-intel-true-view-technology/#gs.vfza9k. [3-August-2019].

[14] Christoph Kieslich. Tageswoche. https://tageswoche.ch/form/kommentar/ein-geniestreich-allein-reicht-nicht/.

[15] Roger Labbe. *Kalman and Bayesian Filters in Python*. 2018.

[16] Labbe, Roger. FilterPy Documentation. https://filterpy.readthedocs.io/en/latest/. [3-August-2019].

[17] Labbe, Roger. Github: Filterpy. https://github.com/rlabbe/filterpy. [3-August-2019].

[18] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[19] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[20] Meta Motion. Magnetic Motion Capture. https://metamotion.com/motion-capture/magnetic-motion-capture-1.htm. [11-August-2019].

[21] Michael Bane. How Hawk-Eye ball tracking can improve tennis performance. https://phys.org/news/2015-01-hawk-eye-ball-tracking-tennis.html. [3-August-2019].

[22] Pedro Nogueira. Motion capture fundamentals a critical and comparative analysis on real-world applications. 2012.

[23] OpenPose. Github. https://github.com/CMU-Perceptual-Computing-Lab/openpose. [3-August-2019].

[24] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V. Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. *CoRR*, abs/1511.06645, 2015.

[25] Basilio Pueo and Jose Jimenez-Olmedo. Application of motion capture technology for sport performance analysis. *Retos: nuevas tendencias en educación física, deporte y recreación*, 2017:241–247, 07 2017.

[26] Konstantinos Rematas, Ira Kemelmacher-Shlizerman, Brian Curless, and Steve Seitz. Soccer on your tabletop. In *CVPR*, 2018.

[27] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[28] Jamie Shotton, Andrew Fitzgibbon, Andrew Blake, Alex Kipman, Mark Finocchio, Bob Moore, and Toby Sharp. Real-time human pose recognition in parts from a single depth image. In *CVPR*. IEEE, June 2011. Best Paper Award.

[29] Kyle Simek. Computer vision blog: The perspective camera - an interactive tour. http://ksimek.github.io/2012/08/13/introduction/, 2013.

[30] Telegraph. Manchester City, Liverpool and Arsenal agree Intel partnership to bring 360-degree replays to Premier League. https://www.telegraph.co.uk/football/2019/02/07/manchester-city-liverpool-arsenal-agree-intel-partnership-bring/. [3-August-2019].

[31] Vizrt. The ultimate tools for live sports production. https://www.vizrt.com/sports. [3-August-2019].

[32] Greg Welch and Gary Bishop. *An Introduction to the Kalman Filter*. 2006.

[33] Wikipedia. Motion capture. https://en.wikipedia.org/wiki/Motion_capture#Inertial_systems. [11-August-2019].

[34] Wikipedia. Random sample consensus (ransac). https://en.wikipedia.org/wiki/Random_sample_consensus.

[35] Wikipedia. Triangulation (computer vision). https://en.wikipedia.org/wiki/Triangulation_(computer_vision). [3-August-2019].

[36] Xsens. Messi in Xsens motion capture suit for FIFA 16. https://www.xsens.com/news/messi-in-xsens-motion-capture-suit-for-fifa-16/. [11-August-2019].

[37] Xsens Technologies B.V. An introduction to the beginning of motion capture technology. https://www.xsens.com/fascination-motion-capture/. [10-August-2019].

[38] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. 11 2015.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Soccer Analysis:
3D Pose Estimation from Television Recordings

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Buner	Tobias

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 22. August 2019	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.