ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Soccer Analysis, 3D Pose Fusion using Extended Kalman Filtering

Bachelor Thesis

Tobias Buner

August 22, 2019

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

Abstract

Contents

Abstract i

Contents iii

List of Figures v

List of Tables vi

1 Introduction 1
- 1.1 Focus of this Work . . . 2
  - 1.1.1 Thesis Organization . . . 2
  - 1.1.2 Outlook on the Method . . . 2

2 Related Work 5
- 2.1 Motion Capture . . . 6
- 2.2 Human Poses Estimation . . . 8
- 2.3 Soccer on Your Tabletop . . . 9
  - 2.3.1 Detectron . . . 10
  - 2.3.2 Calibration . . . 11
  - 2.3.3 OpenPose . . . 12
  - 2.3.4 Depth Map Estimation . . . 14
- 2.4 Filterpy . . . 14

3 Method 17
- 3.1 Calibration . . . 17
  - 3.1.1 Camera Matrix . . . 18
  - 3.1.2 Line Detection . . . 18
- 3.2 Player Detection . . . 18
- 3.3 2D Pose Estimation . . . 18
- 3.4 2D Refinement & mapping . . . 18
- 3.5 3D Pose Fusion . . . 18
  - 3.5.1 Triangulation . . . 18
  - 3.5.2 Extended Kalman Filter . . . 19
  - 3.5.3 Designing the EKF . . . 20

4 Results and Discussion 27

5 Conclusion **29**
5.1 Future Work 29

Bibliography **31**

List of Figures

1.1 Freekick Analysis and Visualization example from Vizrt's sports graphics and analysis tool. 1

2.1 High-definiton 5K cameras locations for the Intel True View technology solution in a soccer stadium. 5

2.2 Messi in Xsens motion capture suit for FIFA 16 to animate his dribbling style as realistically as possible. 7

2.3 Overview for the 3D positions reconstruction from one single depth image. 7

2.4 human 2D pose estimation example with OpenPose 8

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. 9

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. 10

2.7 Numbering of the 18 keypoint of the COCO pose format from *OpenPose*. 12

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). 13

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). 14

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. 17

3.2 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. 19

3.3 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. 20

List of Tables

Chapter 1

Introduction

Nowadays sport is more than what it seems to be. Competitive physical activities at the highest level like Soccer, American Football, Tennis or the Olympic Games count to the most-watched television broadcasts. FIFA announced that more than half the world joined an official broadcast of the World Cup 2018 in Russia [8]. Due to the evolving technologies as high resolution cameras and super computers, it's not surprising that an interesting field of computer visualization has evolved from them.

Tools have been developed for the large number of viewers to bring them closer to the action. Intel True View end-to-end technology solution [11] makes it possible to render 360-degree replays to give the spectator the feeling of standing on the pitch himself. Other tools like the Vizrt's sports graphics and analysis tools [30] also provides data driven augmented reality but also enables the application for advanced analysis.

Figure 1.1: Freekick Analysis and Visualization example from Vizrt's sports graphics and analysis tool.
Source: [30]

This leads to the second large field of application of these technologies. Advanced analysis through Visual Computing in sports is important for teams but also for the referee and the audience. The teams try to improve their tactics based on statistics or to learn from past mistakes. Interested specta-

tors are also fascinated by facts from the statistics such as the number of shots at the goal of both teams in soccer. But it also has an impact on the game. The simplest example is the video assistant referee, he draws the referee's attention to a questionable situation and gives him the opportunity to look at the scene again. Obviously this example does not include any Visual Computing theory but is simply a review of the camera recordings. Another example with computer vision based technology is the Hawk-Eye ball tracking in tennis [19]. 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 they are so called line umpire which are responsible to call the ball out. By the Hawk-Eye system there was a new opportunity for the players, if they think the ball landed in they can challenge the call of the line umpire. This challenge is then performed using the Hawk-Eye system and the result displayed on the screens in the stadium to be visible to all and to provide clarity.

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, thus opening up new possibilities. If, for example, not only the position on the pitch is known for each player, but also a three-dimensional skeleton model, this makes it possible to make a statement about the angle of view of the respective player.

1.1.1 Thesis Organization

The thesis is structured in the following manner. Chapter 2 explains other work and discusses the differences to this approach or the similarities that are used in the same fashion in this work. 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 are some conclusions and what could be further improved in the work.

1.1.2 Outlook on the Method

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

- In order to fit skeleton models to all players in 3D space, stable skeleton joint positions in 2D images had to be estimated with OpenPose. [21].

- The camera calibrations were additionally necessary in order to be able to fuse the 2D positions.
- For the multi-view aggregation the extended Kalman Filter was used.

Chapter 2

Related Work

As mentioned in the Introduction there exists various systems in this area, so the Intel True View end-to-end technology solution [11] is able to render a complete 360-degree replay. The technology behind it is roughly explained in a video [12] as follows: There are about 38 high-definition 5K cameras installed in a stadium processing terabytes of data every minute across 38 different servers. But the cameras generate not just images of two-dimensional pixels, instead they generate massive amount of volumetric data (voxels), pixels with volume, which allows them to capture height, width and depth. This means you do not lose the depth when capturing a moment with the camera and you are able to map the 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.

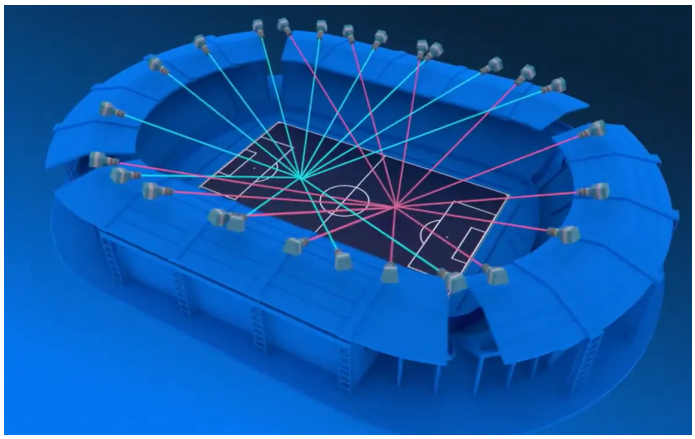

Figure 2.1: High-definiton 5K cameras locations for the Intel True View technology solution in a soccer stadium.
Source: [29]

The swiss national soccer team asked to explore state-of-the-art computer vision and visualization technologies for the data of some tv cameras and not to install an expensive system. So what are the possibilities without installing a dozens of high-definition cameras around the stadium to track the bodies of the players. A large related research are has been developed in the last few decades on motion capture technique.

2.1 Motion Capture

Motion capture tries to track and record the motions of the human body. Whereas the human body is interpreted as a system of rigid links connected by joints as expressed by Xsens [36] 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 and combine several techniques to improve the outcome. The main distinction that is particularly relevant for this work is the difference between marker-based motion capture and markerless motion capture [20].

Marker-Based Motion Capture

Marker-Based Motion Capture can be further categorized into mechanical, magnetic, acoustic and optical [36]. 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 [20].

- Mechanical motion system directly track the joints angles using a exoskeleton worn by the person [32]. 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 field on order to calculate positions and orientations [18, 32, 36]. Magnetic systems do not suffer from line of sight problems, thereby the method is useful if the scene is not in the angle of view.
- Acoustic tracking systems work with ultrasonic pulses and can compute the position through the time-of-flight of the pulses with triangulation. To work correctly a clear line of sight is needed and to avoid reflections this method should be used outdoors [36].
- Optical motion capture uses multiple cameras to track the markers, the markers for this purpose can be passive or active exactly as described above [36]. To extract the markers 3D position triangulation is used ad in the acoustic system. There exist even single camera systems but with additionally sensors to measure the lost depth in the image because triangulation is not possible as noted in [23].

But in general those approaches are all limited to the fact that the player are required to wear some form of markers on their body. Of course it would be possible to place the markers inside the cloths and shoes but that 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 Football 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: [35]

Markerless Motion Capture

Due to the many researches in the field of computer vision, markerless optical motion detection systems are 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 other sensors [23]. An example for such a system is the Microsoft's Kinect [27], they local-

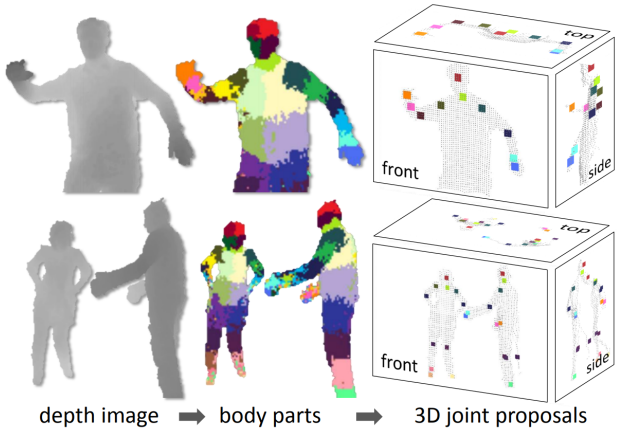

Figure 2.3: Overview for the 3D positions reconstruction from one single depth image.
Source: [27]

ize 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 classification problem per pixel. But for further reading take a look at the paper [27].

2.2 Human Poses Estimation

During the last few years the human 2D pose estimation has been greatly improved. Through better object detection algorithms, as the one presented in the section 2.3.1, common approaches try to detect a person and run for each detection a single-person pose estimation. 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 for multiple people so called buttom-up approaches have been developed as *DeepCut* [22] or *OpenPose* [2]. A rough overview how they work is given in the section 2.3.3 for the *OpenPose* approach.

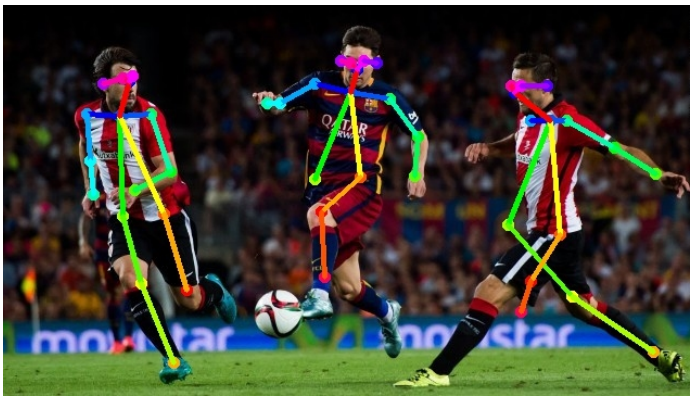

Figure 2.4: human 2D pose estimation example with OpenPose
Source: [3]

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 beginning for 3D human pose estimations.

In order to understand the 3D geometry the depth is needed but exactly this depth is lost in 2D images. To recover the depth from images or videos there exists already comprehensive work such as [10, 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*[22].

Afterwards they fit a statistical body shape model to the 2D joints. The corresponding model is called the *Skinned Multi-Person Linear Model (SMPL)* [17] 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 enables the possibility for triangulation based methods. Along we can benefit from several consecutive frames to not treat each frame individually.

2.3 Soccer on Your Tabletop

This section presents the main concepts of the project *Soccer on Your Tabletop*[25]. The goal of their work is 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*[9]

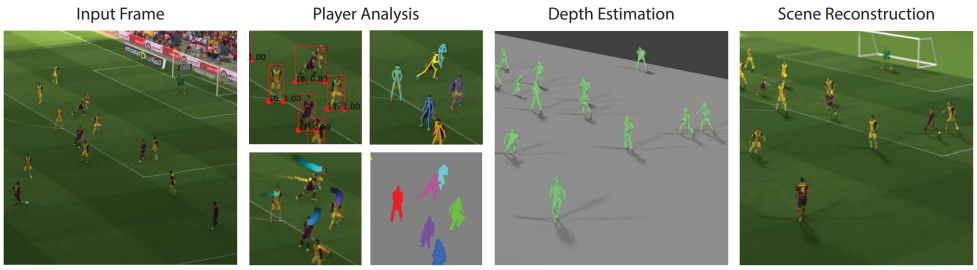

Figure 2.6: Overview of their reconstruction pipeline. In the first step the players from the input frame are analyzed and the camera calibrated. Through the player analysis they segment the player and reconstruct the depth map on the pitch per player. At the end they render the scene in 3D for a complete scene reconstruction.
Source: [25]

is used to detect the persons in each frame. The 2D poses for the recognized persons are then determined using *OpenPose*[2]. These poses together with the detection are then used to refine the people segmentation [37]. For the per player depth map on the pitch they developed a deep network. The network was trained on video game data, to be more precise they extracted depth maps from EA FIFA games. The advantage of this training data is the specific form in terms of movement, clothing and camera perspective [25].

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 section. How they are integrated into our work is then explained in the method chapter.

2.3.1 Detectron

Detectron is a system developed by Facebook AI Research and implements multiple state-of-the-art object detection algorithms such as Mask R-CNN [26], RetinaNet [16] and some others. The system is powered by the Caffe2 deep learning framework which is now part of PyTorch [24].

Soccer on your tabletop modified the originally published *infer_simple.py* file by Detectron which outputs the visualization of the algorithm in PDF format. Instead bounding boxes around the recognized objects and segmentation masks are outputted. 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 and bounding boxes with an area under a specific threshold are removed just to keep the boxes for the players [25]. These are used in later processes.

2.3.2 Calibration

During the calibration step, the camera matrix is determined for each frame of a camera. The camera matrix M transforms homogeneous 3D world coordinates to homogeneous 2D image coordinates and can be decomposed into an intrinsic and extrinsic camera matrix. $\mathbf{R}$ and $\mathbf{T}$ build together the extrinsic camera matrix which describes the camera's position and view direction in world coordinates. If you apply the extrinsic matrix to homogeneous 3D world coordinates, they will be transformed by a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{T}$ to 3D camera coordinates. The intrinsic matrix $\mathbf{A}$ than transforms the 3D camera coordinates to 2D homogeneous screen coordinates. The intrinsic matrix includes the focal length, principal point offsets and the axis skew. The focal length describes the distance between the image plane and the pinhole of the camera. A perpendicular line to the image plane which intersect 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. The axis skew is responsible for the shear distortion which is for a true pinhole camera equal to zero [28].

Manual Initialization

The calibration works in two phases, during the first phase four matching pairs of points are selected manually on the first frame and on a 2D reference image of the football field. A matching pair of points consists of the 2D coordinates for the selected point of the first frame and the 3D coordinates for the selected point on the reference image with z-coordinates equal to zero. With these point pairs the intrinsic camera matrix $\mathbf{A}$ is determined by a grid search. The grid search estimates for different focal lengths the camera matrices using the *solvePnP* function from the OpenCV library [4]. 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 2 under the assumption that the principal axis intersects the image at the center.
The extrinsic camera matrices are then estimated using the matching pair of points from before with the estimated intrinsic matrix using the *solvePnPRansac* function from the same library. It works in the same fashion as the grid search fo find a solution which minimizes the reprojection error. The use of the *Random sample consensus (RANSAC)*[33] 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 are removed which are not the local maximum in the direction of the gradient to consider just 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 [25] project.

2.3.3 OpenPose

Through *Detectron* the location of recognized objects are known 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 a people array of objects including the poses using the *JSON* file format.

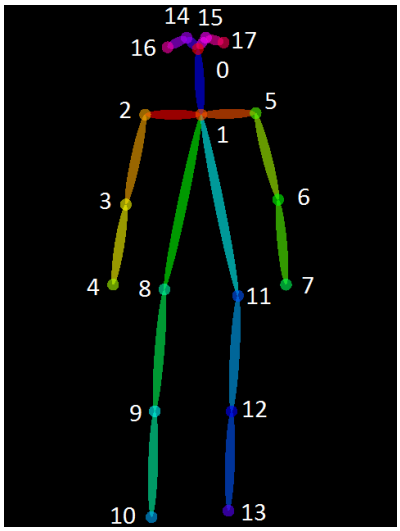

Figure 2.7: Numbering of the 18 keypoint of the COCO pose format from *OpenPose*.
Source: [21]

The pose output format (COCO) was used consisting of 18 keypoints as shown in figure 2.7. In addition to the keypoints coordinates the output also includes so called confidence scores per keypoint in the range $[0, 1]$ to assess the accuracy.

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 persons. So the algorithm predicts body part locations independent of the number of persons in the image and tries to connect the individual body parts so that the result are people. This is the reason for the so called buttom-up approach, but in contrast to other methods such as *DeepCut*[22] the final analysis to map the body parts together is more efficient.

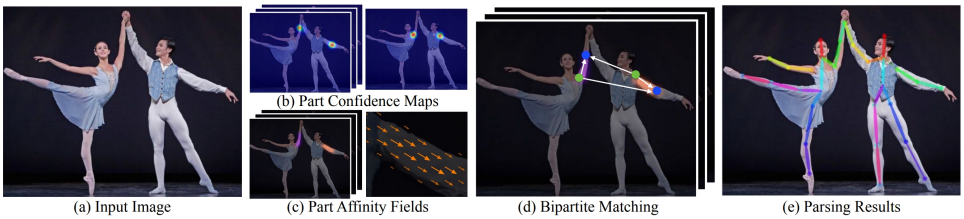

Figure 2.8: *OpenPose* Pipeline Overview. The input image (a) is processed to produce the confidence maps (b) and part affinity fields (c) using a CNN. The result is formulated as a bipartite matching problem (d) in order to solve the part association (e).
Source: [2]

At the begin 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. Each body part has its own confidence map which represents the belief for each pixel that this body part is located there as the figure 2.8 (b) shows. 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 differs 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 suppression in order to not averaging along the different peaks to be able to distinguish between equal body parts that are close to each other.

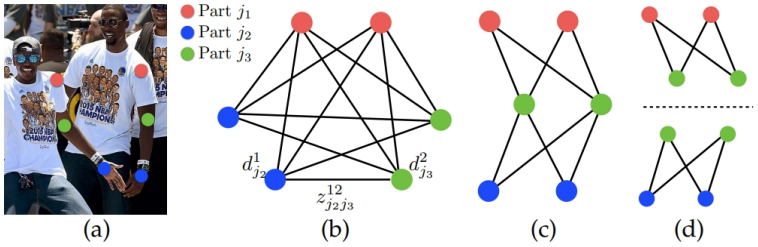

Figure 2.9: *OpenPose's* Part Association formulated as a matching problem with the use of part affinity fields. (a) shows the detected body parts which are then connected in a complete graph (b). The first relaxation reduces the graph to spanning tree skeletons (c) and the second relaxation decompose 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 the detected body part pairs (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. So 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 expresses these so called candidate limbs weighted by the confidence score. So 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 efficient 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 decompose 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

2.4 Filterpy

FilterPy is a Python library that implements a number of Bayesian filter like for example the Extended Kalman Filter. The author of the library also wrote

a book named *Kalman and Bayesian Filter in Python* [13]. Because the free book was written using Ipython Notebook, it offers the perfect interactive guide to deal with bayesian filter. It starts with some simple filters up to the Kalman filter. It describes the implementation of the library with many examples and explains the mathematical background to understand why it works. For more information visit the the github page [15].

Chapter 3

Method

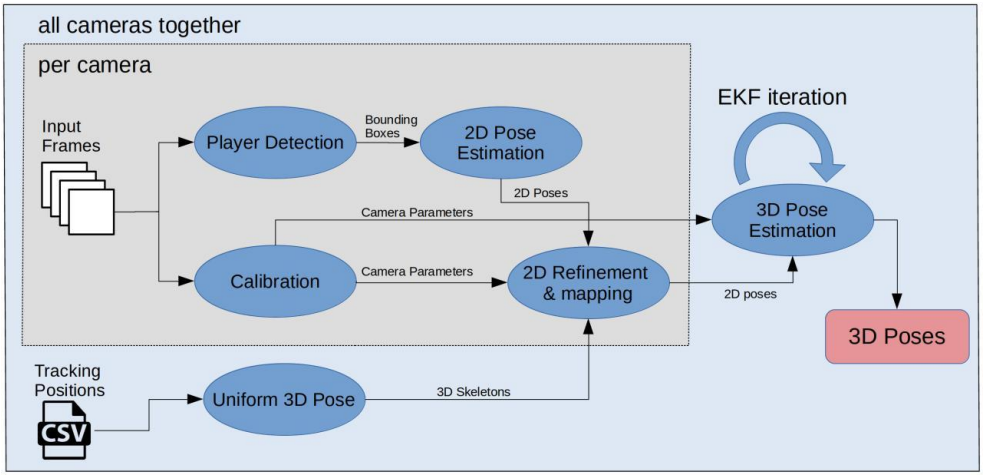

Figure 3.1: Overview of the workflow. Each step gets described in detail during this chapter. The gray area is the process for each camera separately while in the blue area the different cameras are merged.

3.1 Calibration

In order to work in three-dimensional space it is required to know how the pixels in a two-dimensional image relate 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 M which transforms homogeneous 3D world coordinates to homogeneous 2D image coordinates.

3.1.1 Camera Matrix

3.1.2 Line Detection

3.2 Player Detection

3.3 2D Pose Estimation

3.4 2D Refinement & mapping

3.5 3D Pose Fusion

The next goal is to determine the points in 3D space given its projections onto the multiple images from the different camera perspectives. By knowing the camera matrix of the camera projection from 3D to 2D 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 2D image as mentioned in section 3.1.

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 3.2 [34]. The lines intersect at x which is the corresponding point in 3D space.
But in practice the the image points are measured with arbitrary accuracy. This leads to the two points y_1' and y_2' in the figure 3.2. If you consider the corresponding lines of these two points in 3D space (blue), they must 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 camera calibration which lead to wrong focal lengths O_1 and O_2 and therefore to a wrong projection [34].

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 the camera calibration is not error-free, I 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 be able to get the 3D poses.

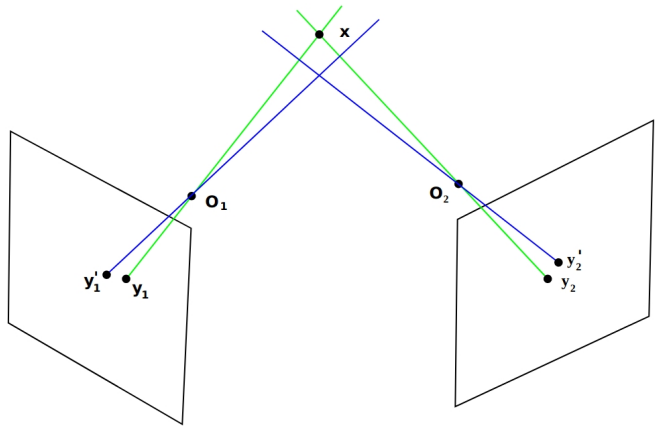

Figure 3.2: Triangulation visualization of two images with the camera's focal length O_1 and O_2. The ideal case with perfect measurements in green and the case of measurements with arbitrary accuracy in blue.
Source: [34]

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* [31] and *Kalman and Bayesian Filters in Python* [13].

The algorithm works in a two-step process, as shown in figure 3.3, which is repeated for every time step or more precise 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 variable gets updated using the measurement residual weighted with the so called Kalman gain. Additionally the error covariance is updated. and a new external measurement as described in section x.

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.

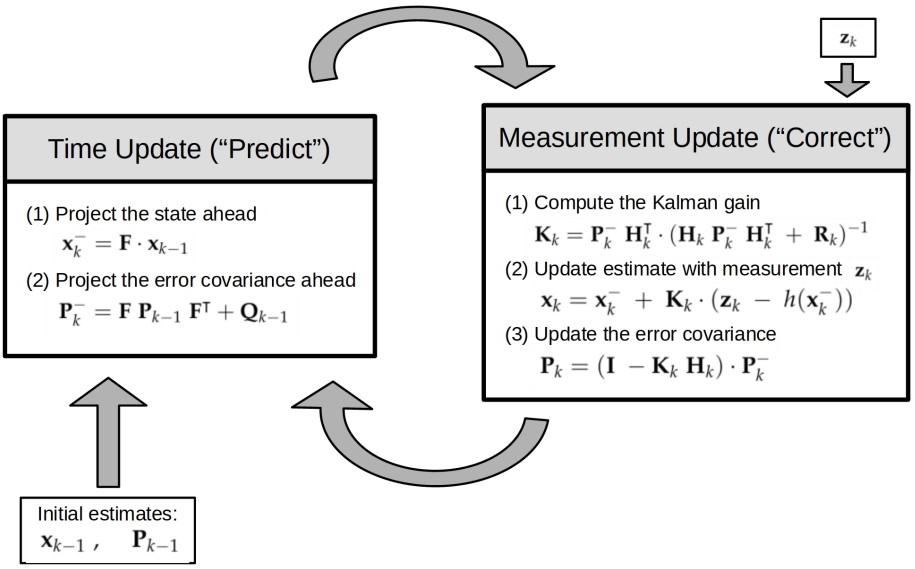

Figure 3.3: Complete picture of the calculations during the two-step process of the Extended Kalman Filter. For the Initialization the values of the first state vector and the error covariance are needed. A new measurement vector is given to the filter for each Measurement Update step. **Source:** Notation slightly adapted from [31]

3.5.3 Designing the EKF

In the figure 3.3 and in the following text the notation from the book *An Introduction to the Kalman Filter* [31] slightly adapted to the filterpy [13] 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 2D 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 15 keypoints. For the i-th keypoint the corresponding part of the state vector includes the position in 3D space and their velocities:

$$\mathbf{x}^i := \begin{bmatrix} x_w^i & y_w^i & z_w^i & v_x^i & v_y^i & v_z^i \end{bmatrix}^\mathsf{T}, \quad \forall i \in \{0, 1, ..., 17\} \tag{3.1}$$

All 18 keypoints together form the state vector:

$$\mathbf{x} := \begin{bmatrix} \mathbf{x}^0 & \mathbf{x}^1 & \cdots & \mathbf{x}^{17} \end{bmatrix}^\mathsf{T} \tag{3.2}$$

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

$$\mathbf{z}^i := \begin{bmatrix} x^i_{c_0} & y^i_{c_0} & x^i_{c_1} & y^i_{c_1} & \cdots & x^i_{c_n} & y^i_{c_n} \end{bmatrix}^\mathsf{T}, \quad \forall i \in \{0, 1, ..., 17\} \tag{3.3}$$

The complete measurement vector is again for all 18 keypoints:

$$\mathbf{z} := \begin{bmatrix} \mathbf{z}^0 & \mathbf{z}^1 & \cdots & \mathbf{z}^{17} \end{bmatrix}^\mathsf{T} \tag{3.4}$$

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 differently 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' propagates the state vector for the i-th keypoint x^i further in time,

$$\mathbf{F}^i := \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \forall i \in \{0, ..., 17\} \tag{3.5}$$

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

$$\mathbf{F} := \begin{bmatrix} \mathbf{F}^1 & & \\ & \ddots & \\ & & \mathbf{F}^{17} \end{bmatrix} \tag{3.6}$$

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

$$\mathbf{x}_k^- = \mathbf{F} \cdot \mathbf{x}_{k-1} \tag{3.7}$$

Together with the next equation 3.8, 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.

$$\mathbf{P}_k^- = \mathbf{F}\ \mathbf{P}_{k-1}\ \mathbf{F}^\mathsf{T} + \mathbf{Q}_{k-1} \tag{3.8}$$

At the begin the covariance matrix $\mathbf{P}$ have 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 3.8 and 3.11 we add a covariance matrix 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 $\sigma_q = 0.3$ and $\sigma_r = 0.02$:

$$\mathbf{Q} := \begin{bmatrix} \sigma_q & & \\ & \ddots & \\ & & \sigma_q \end{bmatrix} \qquad \mathbf{R} := \begin{bmatrix} \sigma_r & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix} \tag{3.9}$$

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 a 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:

$$\mathbf{K}_k = \mathbf{P}_k^- \, \mathbf{H}_k^\intercal \cdot (\mathbf{H}_k \, \mathbf{P}_k^- \, \mathbf{H}_k^\intercal \, + \, \mathbf{R}_k)^{-1} \tag{3.10}$$

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

$$\mathbf{P}_k = (\mathbf{I} \, - \mathbf{K}_k \, \mathbf{H}_k) \cdot \mathbf{P}_k^- \tag{3.11}$$

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 [31]. These two equations involves the matrix $\mathbf{H}$ which is explained in the section 3.5.3 below.

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:

$$\mathbf{x}_k = \mathbf{x}_k^- \; + \; \mathbf{K}_k \cdot (\mathbf{z}_k \; - \; h(\mathbf{x}_k^-)) \tag{3.12}$$

The explanation how the function h looks like and what the related matrix $\mathbf{H}_x$ is will follow in detail. The filterpy implementation [14] 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 do not get confused, it is just the same space and no new measurement vector:

$$h(\mathbf{x}_k^-, \text{camera matrices}) = \mathbf{z}_{transformed} \tag{3.13}$$

As mentioned in section 3.1.1 a homogeneous point in the 3D space can be projected to the image screen using the camera matrix $\mathbf{M}$. I 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. I will omit the time subscript k in this section for the sake of understanding.

$$\mathbf{M}_{c_j} \cdot \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \mathbf{A}_{c_j} \begin{bmatrix} \mathbf{R}_{c_j} & | & \mathbf{T}_{c_j} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3.14}$$

$$= \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_x \\ r_{2,1} & r_{2,2} & r_{2,3} & t_y \\ r_{3,1} & r_{3,2} & r_{3,3} & t_z \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3.15}$$

$$= \begin{bmatrix} x'_{c_j} \\ y'_{c_j} \\ z'_{c_j} \end{bmatrix} \tag{3.16}$$

In order to get the correct scale and dimension for the screen coordinates x_{c_j}, y_{c_j} one have to divide the result from equation 3.16 by z_{c_j} and drop the

last entry for the z coordinate. This forms a part of the function h:

$$\hat{h}\left(\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix}, \mathbf{A}_{c_j}, \mathbf{R}_{c_j}, \mathbf{T}_{c_j}\right) = \begin{bmatrix} x_{c_j} \\ y_{c_j} \end{bmatrix} \tag{3.17}$$

$$= \begin{bmatrix} \frac{x'_{c_j}}{z'_{c_j}} \\ \frac{y'_{c_j}}{z'_{c_j}} \end{bmatrix} = \begin{bmatrix} \frac{f_x \cdot (r_{1,1} \cdot x_w + r_{1,2} \cdot y_w + r_{1,3} \cdot z_w + t_x)}{r_{3,1} \cdot x_w + r_{3,2} \cdot y_w + r_{3,3} \cdot z_w + t_z} + x_0 \\ \frac{f_y \cdot (r_{2,1} \cdot x_w + r_{2,2} \cdot y_w + r_{2,3} \cdot z_w + t_x)}{r_{3,1} \cdot x_w + r_{3,2} \cdot y_w + r_{3,3} \cdot z_w + t_z} + y_0 \end{bmatrix} \tag{3.18}$$

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:

$$h_i\left(\begin{bmatrix} x_w^i \\ y_w^i \\ z_w^i \end{bmatrix}, \text{ camera matrices}\right) = \begin{bmatrix} \hat{h}\left(\begin{bmatrix} x_w^i \\ y_w^i \\ z_w^i \end{bmatrix}, \mathbf{A}_{c_0}, \mathbf{R}_{c_0}, \mathbf{T}_{c_0}\right) \\ \vdots \\ \hat{h}\left(\begin{bmatrix} x_w^i \\ y_w^i \\ z_w^i \end{bmatrix}, \mathbf{A}_{c_n}, \mathbf{R}_{c_n}, \mathbf{T}_{c_n}\right) \end{bmatrix} \tag{3.19}$$

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

$$h(\mathbf{x}_k^-, \text{camera matrices}) = \begin{bmatrix} h_0\left(\begin{bmatrix} x_w^0 \\ y_w^0 \\ z_w^0 \end{bmatrix}, \text{ camera matrices}\right) \\ \vdots \\ h_{17}\left(\begin{bmatrix} x_w^{17} \\ y_w^{17} \\ z_w^{17} \end{bmatrix}, \text{ camera matrices}\right) \end{bmatrix} \tag{3.20}$$

$$= \mathbf{z}_{transformed} \tag{3.21}$$

Jacobian Matrix H

By 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 $\mathbf{x}_k$ like a Taylor series. The resulting matrix is the Jacobian matrix $\mathbf{H}_k$. How the derivations of one screen coordinate pair (the i-th keypoint) like in equation 3.17 for the camera

j looks like is explained as follows:

$$\mathbf{H}_{c_j}^i = \begin{bmatrix} \frac{\partial x_{c_j}^i}{\partial x_w^i} & \frac{\partial x_{c_j}^i}{\partial y_w^i} & \frac{\partial x_{c_j}^i}{\partial z_w^i} & \frac{\partial x_{c_j}^i}{\partial v_x^i} & \frac{\partial x_{c_j}^i}{\partial v_y^i} & \frac{\partial x_{c_j}^i}{\partial v_z^i} \\ \frac{\partial y_{c_j}^i}{\partial x_w^i} & \frac{\partial y_{c_j}^i}{\partial y_w^i} & \frac{\partial y_{c_j}^i}{\partial z_w^i} & \frac{\partial y_{c_j}^i}{\partial v_x^i} & \frac{\partial y_{c_j}^i}{\partial v_y^i} & \frac{\partial y_{c_j}^i}{\partial v_z^i} \end{bmatrix} \tag{3.22}$$

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:

$$\mathbf{H}^i = \begin{bmatrix} \mathbf{H}_{c_0}^i \\ \mathbf{H}_{c_1}^i \\ \vdots \\ \mathbf{H}_{c_n}^i \end{bmatrix} \tag{3.23}$$

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.

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{H}^0 & & \\ & \ddots & \\ & & \mathbf{H}^{17} \end{bmatrix} \tag{3.24}$$

Like the function h the filterpy [14] 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:

$$r_1\mathbf{x}^i \coloneqq r_{1,1} \cdot x_w^i + r_{1,2} \cdot y_w^i + r_{1,3} \cdot z_w^i \tag{3.25}$$

$$r_2\mathbf{x}^i \coloneqq r_{2,1} \cdot x_w^i + r_{2,2} \cdot y_w^i + r_{2,3} \cdot z_w^i \tag{3.26}$$

$$r_3\mathbf{x}^i \coloneqq r_{3,1} \cdot x_w^i + r_{3,2} \cdot y_w^i + r_{3,3} \cdot z_w^i \tag{3.27}$$

So what's left are the partial derivatives. The computation for those from equation 3.22 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.

$$\frac{\partial x_{c_j}^i}{\partial x_w^i} = \frac{f_x \cdot r_{1,1}}{r_3 \mathbf{x}^i + t_z} - \frac{f_x \cdot r_{3,1}(r_1 \mathbf{x}^i + t_x)}{(r_3 \mathbf{x} + t_z)^2} \tag{3.28}$$

$$\frac{\partial x_{c_j}^i}{\partial y_w^i} = \frac{f_x \cdot r_{1,2}}{r_3 \mathbf{x}^i + t_z} - \frac{f_x \cdot r_{3,2}(r_1 \mathbf{x}^i + t_x)}{(r_3 \mathbf{x} + t_z)^2} \tag{3.29}$$

$$\frac{\partial x_{c_j}^i}{\partial z_w^i} = \frac{f_x \cdot r_{1,3}}{r_3 \mathbf{x}^i + t_z} - \frac{f_x \cdot r_{3,3}(r_1 \mathbf{x}^i + t_x)}{(r_3 \mathbf{x} + t_z)^2} \tag{3.30}$$

$$\frac{\partial y_{c_j}^i}{\partial x_w^i} = \frac{f_y \cdot r_{2,1}}{r_3 \mathbf{x}^i + t_z} - \frac{f_y \cdot r_{3,1}(r_2 \mathbf{x}^i + t_y)}{(r_3 \mathbf{x} + t_z)^2} \tag{3.31}$$

$$\frac{\partial y_{c_j}^i}{\partial y_w^i} = \frac{f_y \cdot r_{2,2}}{r_3 \mathbf{x}^i + t_z} - \frac{f_y \cdot r_{3,2}(r_2 \mathbf{x}^i + t_y)}{(r_3 \mathbf{x} + t_z)^2} \tag{3.32}$$

$$\frac{\partial y_{c_j}^i}{\partial z_w^i} = \frac{f_y \cdot r_{2,3}}{r_3 \mathbf{x}^i + t_z} - \frac{f_y \cdot r_{3,3}(r_2 \mathbf{x}^i + t_y)}{(r_3 \mathbf{x} + t_z)^2} \tag{3.33}$$

$$\frac{\partial x_{c_j}^i}{\partial v_x^i} = \frac{\partial x_{c_j}^i}{\partial v_y^i} = \frac{\partial x_{c_j}^i}{\partial v_z^i} = \frac{\partial y_{c_j}^i}{\partial v_x^i} = \frac{\partial y_{c_j}^i}{\partial v_y^i} = \frac{\partial y_{c_j}^i}{\partial v_z^i} = 0 \tag{3.34}$$

State Initialization

TODO

All together result in the final EKF. For a good overview the five equations for the filter are repeated here:

The Update ("Predict"):

(1) Projet the state ahead:

$$\mathbf{x}_k^- = \mathbf{F} \cdot \mathbf{x}_{k-1} \tag{3.35}$$

(2) Project the error covariance ahead:

$$\mathbf{P}_k^- = \mathbf{F}\ \mathbf{P}_{k-1}\ \mathbf{F}^\intercal + \mathbf{Q}_{k-1} \tag{3.36}$$

Measurement Update ("Correct"):

(1) Compute the Kalman gain:

$$\mathbf{K}_k = \mathbf{P}_k^-\ \mathbf{H}_k^\intercal \cdot (\mathbf{H}_k\ \mathbf{P}_k^-\ \mathbf{H}_k^\intercal\ +\ \mathbf{R}_k)^{-1} \tag{3.37}$$

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

$$\mathbf{x}_k = \mathbf{x}_k^-\ +\ \mathbf{K}_k \cdot (\mathbf{z}_k\ -\ h(\mathbf{x}_k^-)) \tag{3.38}$$

(3) Update the error covariance:

$$\mathbf{P}_k = (\mathbf{I}\ -\mathbf{K}_k\ \mathbf{H}_k) \cdot \mathbf{P}_k^- \tag{3.39}$$

Chapter 4

Results and Discussion

Chapter 5

Conclusion

5.1 Future Work

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

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

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

[12] 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].

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

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

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

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

[17] 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.

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

[19] 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].

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

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

[22] 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.

[23] 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.

[24] PyTorch. Github. https://github.com/pytorch/pytorch.

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

[26] 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.

[27] 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.

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

[29] 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].

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

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

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

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

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

[35] 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].

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

[37] 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):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

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

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)**

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