DETECTING OFFSIDE POSITION USING 3D RECONSTRUCTION

DETEKTION AV OFFSIDE POSITION GENOM 3D REKONSTRUKTION

AXEL DELBOM

Master's thesis
2020:E74

LUND UNIVERSITY

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Abstract

With the increased number of technical innovations in soccer and the progress in the field of AI, this thesis investigates the possibilities to detect if a player is in offside position using deep learning and video recordings from soccer games. A method to create a 3D reconstruction of the players in the scene is proposed and evaluated. The arguments for using a 3D reconstruction is that it introduces a volume to the players which hopefully result in increased quality of offside decisions. A system for detecting offside position is proposed consisting of an object detector, a pose estimator, a mesh estimator and a team classification. The position of the players are estimated by minimizing the reprojection error between the 2D pose and the 3D pose, further on the reprojection is weighted using the confidence of each predicted 2D joint. The system is able to detect players in offside position with a precision of 60% and a recall of 97%. It is also shown that by weighting the reprojection error the precision of detected players in offside position is increased from 59% to 60%. It is also shown that reconstructing the scene increases the average F1 score from 0.82 to 0.86.

Acknowledgements

I would like to thank my supervisor Mikael Nilsson at Lunds University for helpful discussions and feedback. I would also like to thank Ann-Sofie Rase, Johan Olsson, Daniel Larsson and everybody else at Axis Communications for useful feedback and being available to answer my questions. Thanks also to Spiideo for providing recorded soccer games.

Contents

Chapter 1
Introduction

1.1 Motivation

The international soccer has opened up for technical solutions to improve the sport. In the latest years there have been several innovations such as goal line technology and Video Assistant Referee (VAR). VAR have been introduced with hope to correct incorrect decisions by the referee, a group of referees analyze videos of the game in real time to find incorrect decisions. VAR has been criticized where criticizers believe that video assessment destroys the game, interruptions is introduced when decisions are revised be the video referee. Classification of whether a player is in offside position should be possible to make using image analysis. The thesis tries to solve this problem using a monocular camera since it provides a cheaper and more available system.

1.2 Aim of the thesis

The aim of the thesis is to investigate if players that are in offside position can be detected using a monocular camera. The thesis also investigates if the performance of the classifier can be improved by creating a 3D reconstruction with the help of deep learning.

Chapter 2
Theory

2.1 Artificial Neural Network

Artificial neural network abbreviated as ANN is a graph system inspired by the biological nervous system. An ANN is a graph of connected neurons where each connection can transmit a signal between neurons much like synapses. Using learning examples, learning data, a network learns to perform a specific task such as image recognition.

2.1.1 Perceptron

The perceptron is the simplest form of neuron consisting of a single non-linear neuron. It was introduced by F.Rosenblatt [1] in 1958 as a model of a trainable neuron to classify some input into two classes. A perceptron has an input $x \in \mathbb{R}^D$ also called features $[x_1, x_2, \dots, x_D]$ with corresponding weights $[w_1, w_2, \dots, w_D]$, a bias b and some activation function ϕ. The output of a perceptron is defined as

$$y = \phi(\sum_{i}^{N}(w_i x_i) + b). \tag{2.1}$$

The described model can be viewed in figure 2.1. In the original perceptron the activation function was a threshold function sgn defined as,

$$sgn(a) = \begin{cases} a < 0 & -1 \\ a \geq 0 & 1 \end{cases}, \tag{2.2}$$

where the bias is used to adjust the threshold limit. The bias and weights are adjustable to achieve the desired output from the perceptron. Often the bias is expressed by adding an extra feature to the input vector with a constant value of one. The single perceptron is capable of separating two classes if they are linearly separable i.e the classes can be separable by a hyperplane.

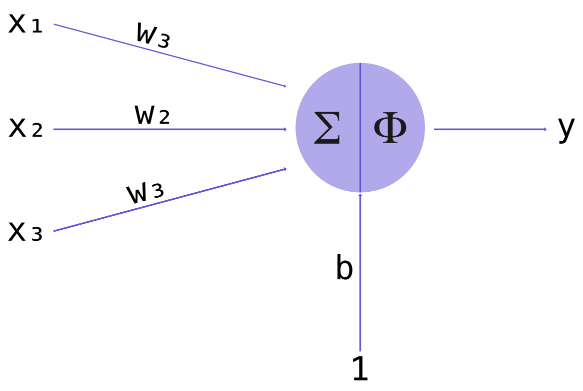


Figure 2.1: Graphical model of the perceptron described in the section above

2.1.2 Multilayer Perceptron

The multilayer perceptron is as the name indicates a network built by multiple layers of perceptrons. The first layer in the network is called the input layer and the final layer is called the output layer. Layers in between the first and final layers are called hidden layers. The construction of the multilayer perceptron is shown in figure 2.2. The multilayer perceptron is also called feedforward network. The name arise from how information flows in the network, from the input layer to the output layer without any back edges. This means that the output from a neuron is connected to the input of one or more neurons in a succeeding layer. The graph formed by the feedforward network is therefore a directed acyclic graph [2].

The purpose of the multilayer perceptron is the same as in in the single perceptron to map an input x to a desired output y. However, a multilayer perceptron can create decision boundaries with arbitrary shape by combining multiple hyperplanes hence the targets does not need to be linearly separable.

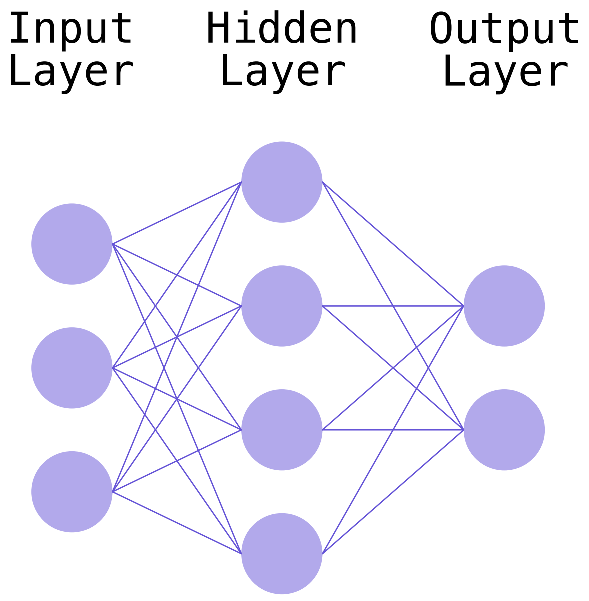


Figure 2.2: Graph of the multilayer perceptron

2.1.3 Training

Weights need to be adjusted to get the desired output given an input x. The process of finding the optimal weights is the training or learning phase of developing neural networks. Optimization such as minimization and maximization are used to find appropriate values for the weights. The function which is optimized is called the objective function f and when minimization is used it is often referred to as loss function. The process of minimization can be expressed as $\underset{\theta}{argmin}\ f(\theta)$ where θ are the weights and biases [2]. The standard method for minimizing an objective function is the gradient descent method. Gradient descent depends on the first order derivative hence the objective function must be differentiable.

2.1.4 Gradient Descent

As mentioned earlier the most commonly used method for optimization in neural networks is gradient descent. Gradient descent minimizes the objective function $f(\theta)$ by updating θ in the opposite direction of the gradient $\underset{w.r.t\ \theta}{\nabla f(\theta)}$. By iteratively updating θ in the opposite gradient direction the algorithm iteratively takes steps towards a local minimum of the objective function. The update rule for gradient descent defined as,

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t), \tag{2.3}$$

where the learning rate η defines with which step size the algorithm moves towards a local minimum [3].

There are different variants of the gradient descent which mainly differs in how much of the data is used in one update iteration. Vanilla gradient descent, also called batch gradient descent, use the complete dataset when computing the gradients. Since the gradients are calculated for the entire dataset for every iteration this method can be slow. It is also a problem if the dataset is large since the entire dataset needs to be loaded to memory. Stochastic gradient descent is the opposite of batch gradient descent as it performs an update iteration for every data point $x^i,\ y^i$ in the dataset. Stochastic gradient descent is generally faster than batch gradient descent since it update parameter θ for every data point and therefore avoid calculating gradients for redundant data points for one update iteration. As the parameters are updated for every data point there will be a larger fluctuation in the gradients, therefore stochastic gradient descent can jump to a new local minima potentially better than the earlier local minima. The fluctuation behavior also complicates the convergence to the exact minimum as the fluctuation might result in parameters which jumps around the minima. The most commonly used approach in training of neural network is mini-batch gradient descent. It is a compromise between the two earlier techniques as it uses a small batch of the dataset to update the parameters. The batch size is generally between 50 to 256 [3]. By using a small subset of the dataset, the variance of the gradients will be lower which can result in a more stable convergence. The size of the mini-batch can be adjusted to fit in memory and therefore matrix optimizations can be utilized, which gives an efficient computation of gradients.

2.1.5 Convolutional Neural Network

Convolutional neural networks often abbreviated as CNN is a variant of ANN most commonly used for processing time-series data and image data. The following section will give an overview of the convolutional layer in a CNN.

2.1.6 Convolution

The convolution operator is an operation on two functions and is defined as,

$$s(t) = \int_{-\infty}^{\infty} x(t)w(t-\tau)d\tau, \tag{2.4}$$

where x is denoted as the input and w is often called kernel. Convolution operation can be used to smooth a signal by taking a weighted average of the input over time. If x denotes the input signal and w is a valid density function the convolution of x and w would result in a new signal $s(t)$ which would be smoothed by the weighted average. When working with digital signals the data is discrete and the definition of the convolution is in the discrete case defined as

$$s(t) = \sum_{\tau=-\infty}^{\infty} x(t)w(t-\tau). \tag{2.5}$$

Convolutions are used to apply filters to images. An image is often represented as a multidimensional array hence the kernel is also multidimensional. In the simplest case where an image only contains one channel an image is represented by a two-dimensional array. The kernel is represented as a matrix which is slid over the image and a new value is calculated for every pixel. A simple example of the convolution operation on an image is shown in figure 2.3 and the result of applying the filter is shown in figure 2.4.

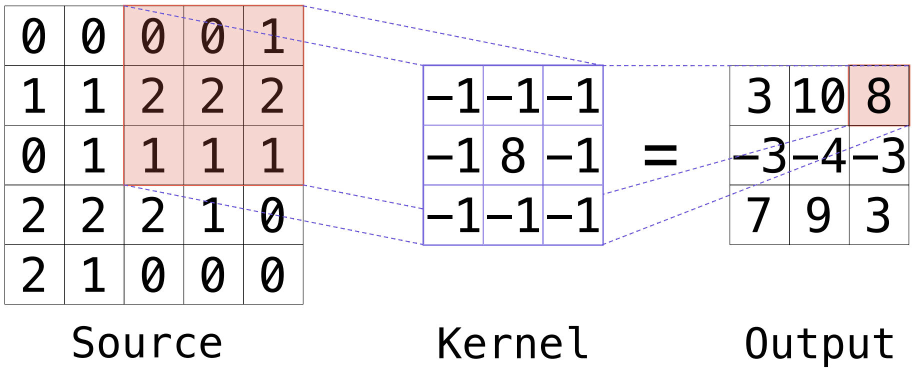


Figure 2.3: A convolutional operation applied on an image and how the source and kernel relates to the output

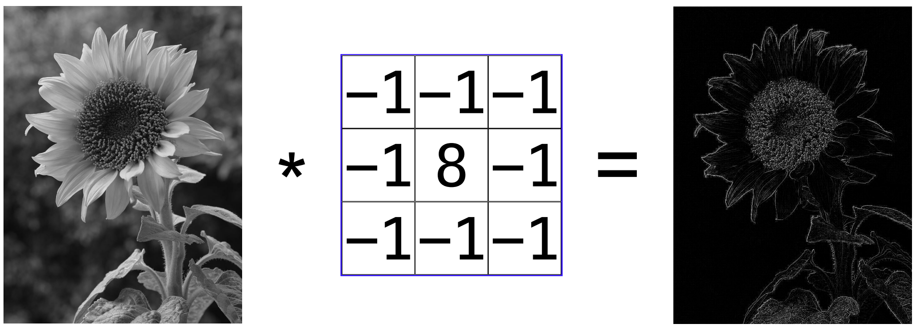


Figure 2.4: Convolution applied to an image with a kernel which is used for edge detection

2.1.7 Convolutional Layer

A convolutional layer is a layer in a neural network which represents a convolution filter. The convolutional layer is sparsely connected if the kernel is smaller than the input which is almost always the case when processing images. Sparse connectivity means that a node only has connections to a subset of the nodes in the succeeding layer which differs from a dense layer where a node is connected to all nodes in the succeeding layer. By being sparsely connected the number of parameters is largely reduced. In a fully connected dense layer there are mxn weights where m is the number of inputs and n the number of outputs. In a convolutional layer the number of weights is reduced to kxn where k is the number of connections to an output node. In the case with a 3x3 kernel one can see that the number of weights is largely reduced. This also affects the performance of the network since there will only be kxn multiplications rather than mxn multiplications. By using convolutional layers instead of dense layers, the size of the network is reduced as well as the number of operations. Convolutional layers utilize the idea of parameter sharing since the weights of the kernel is reused when computing the output from different input nodes. This differs from the traditional layer where one weight is used once per connection between an input and output node. By reusing the parameters, the size of the network is further reduced and the layer is represented by k parameters instead of kxm. Parameter sharing also gives a property called equivariance to translation which means that if the input is translated the output is translated in the same way [2]. Parameter sharing and sparse connection between two layers is shown in figure 2.5. Parameter sharing and equivariance to translation is useful when working with images, if the layer is trained to detect a feature in an image it can detect this feature regardless of the object's translation.

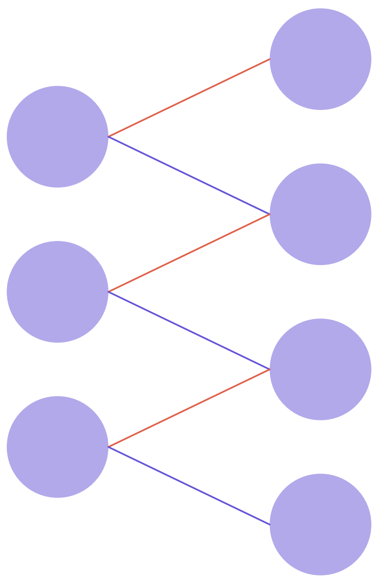


Figure 2.5: A connection between two layers which are sparsely connected and parameter sharing is used. As seen, each node to the left is only connected to two nodes on the right. The color of the edges represents a weight. There are two weights in use which is reused between the nodes.

2.2 Object Detection

There are several deep learning architectures to perform object detection such as Faster-RCNN [4], RetinaNet [5] and YOLO [6]. When choosing architecture, there is a balance between inference speed and precision. This thesis will focus on YOLO as object detection architecture since it has a low inference time and at the same time, an acceptable precision. YOLO is an abbreviation of *You Only Look Once* which is a description of how YOLO performs object detection. Object detectors often use a two-stage process where the detector first find regions of interest, regions where it might be an object. In the second stage each region of interest is classified as one of the predefined classes. YOLO differs from this approach as it finds regions of interest and classification in the same stage. By doing both tasks in one stage YOLO is able to have a greater throughput than networks which use the two-stage pipeline. In this work the focus will be YOLOv3 which at the time is the latest version of YOLO.

2.2.1 Darknet

YOLOv3 use a feature extractor called Darknet 53, as the name suggest it consists of 53 convolutional layers. The feature extractor also takes advantage of residual blocks which means that a layer does not only feed its output to the next layer but also to a layer 2-3 steps ahead. Residual blocks have shown to be useful when constructing deep convolutional network, having a positive effect on ease of optimization and increased accuracy [7]. The purpose of the feature extractor is to reduce the dimensionality of input data, in this manner creating a compressed representation of data.

2.2.2 Detection

Assuming that the feature extractor outputs a feature map of size 13x13xN, where N is the number of kernels of the convolutional layer, there are 13x13 grid cells and YOLO makes predictions for each grid cell. In YOLOv3 the feature map is also upsampled to two larger scales resulting in predictions at three different scales. A grid cell in an upsampled feature map will represent a smaller region of the original image making it easier to detect smaller objects. A common concept in object detection is the use of anchor boxes which also YOLO utilizes. Rather than estimating position and size YOLO starts from anchor boxes where the size is predefined as p_w, p_h. The starting position of the bounding box is the offset from the top left corner to the center of the grid cell denoted as c_x, c_y. The priors of the anchor boxes are calculated by using k-means to cluster the ground truth bounding boxes to 3 clusters per scale resulting in 9 clusters. By estimating t_x, t_y, t_w and t_h the produced bounding box is expressed as

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h}, \end{aligned} \tag{2.6}$$

where σ denotes the sigmoid function. In addition to predicting translation and scaling the network predicts a class score and confidence score per grid cell. The class score is a vector with a score for each object classes. The confidence score is a value between 0 and 1, a high sore indicates that an anchor box contains an object. Since there will be a lot of bounding boxes one does simply filter the list of bounding boxes by thresholding the confidence level by some value.

2.2.3 Non-Maximum Suppression

Even though the number of bounding boxes is greatly reduced by filtering with regard to the confidence score there will most likely be some overlapping bounding boxes which contains the same object. Non max suppression(NMS) is a greedy algorithm which reduces the number of bounding boxes by removing redundant boxes [8]. A box is considered redundant if the overlap is larger than some threshold. The measure of overlap is intersection of union which is defined as

$$IoU = \frac{area\ of\ overlap}{area\ of\ union}. \tag{2.7}$$

The algorithm first chooses the bounding box with the largest confidence score from the list of proposed bounding boxes. All boxes with an IoU between itself and the chosen box larger than some threshold are suppressed. This process is repeated until the list of proposed bounding boxes are empty. There is also another variant of the algorithm called soft non max suppression(soft-NMS) [9] which reduces the confidence score of overlapping boxes with regard to the IoU rather than direct suppression. The advantage of soft-NMS is that there might be two objects located closely to each other which both have boxes with high confidence score. With the NMS approach only the box with the highest score would be kept whilst soft-NMS would decrease the confidence and potentially keep both boxes resulting in a detection of both objects. Soft-NMS have been shown to increase the mean average precision of object detections.

2.3 Histogram Intersection

A very simple model for image classification is the histogram intersection model. Given an image in some color space a histogram is calculated by counting the occurrence of each color in the image and normalized by the number of pixels. Histogram models are invariant to translation and rotation since the histogram does not account for where in an image a color occurs. Two colors are considered equal if they are inside of the same bin in the histogram. The intersection between two histograms is the measurement of similarity between two histograms I and M with n bins, the intersection is defined as

$$\sum_{j=1}^{n} min(I^j, M^j). \tag{2.8}$$

To get the most accurate result it is preferable to remove the background when calculating the histogram and intersection [10]. Each class is represented by a histogram model M, a histogram of an image H is classified as class c if M_i and H has the largest intersection. The classification can be expressed as:

$$\underset{c}{argmax} \sum_{j=1}^{n} min(H^j, M_c^j) \tag{2.9}$$

2.4 Human Pose Estimation

Human pose estimation is a well-studied problem in computer vision. The goal of human pose estimation is to identify body parts in an image. There are two types of human pose estimation, single person and multiple persons.

2.4.1 Single Person

A pose is constructed by identifying key points for a predefined set of joints. In early work, a pose was represented by a stickman with a line for different body parts. Using body joints has the advantage of being more accurate since it only needs to identify a point in the image. Joints are also naturally connected and a stickman can be constructed by interpolating lines between the detected joints. In human pose estimation deep learning has heavily been used to locate the position of key joints. There are mainly two approaches to locate key-points, direct regression of key-points and regression of a heat-map which contains a confidence score of whether a body joint is present in the pixel. Direct regression of body joints in an image is a nonlinear problem, it is therefore difficult to find a direct mapping from an image to key-points [11]. The precision of key-points location from a heat-map depends on the resolution of the produced heat-map. Depending on the resolution, the memory usage of a heat-map solution might be large and is clearly larger than the memory needed for the result from direct regression.

2.4.2 Multiple Persons

Human pose estimations of multiple persons in an image is a more difficult task than a single person pose estimation mainly since the number of humans is unknown. In human pose estimation for multiple persons there are two problems which need to be solved, human detection and key-point localization. There are two approaches for solving multi-person pose estimation which can be categorized in top-down and bottom-up. The top-down approach first detects humans and producing bounding boxes, then run a single person pose estimation for every detected bounding box. The bottom-down approach works in reverse order. It first detects all key-points and then creates a pose per person by matching the detected key-points to a full pose. The result from the top-down approach heavily depends on the quality of the human detector. Since most single person pose estimation networks are trained with images where the person is located in the center of the image, therefore sensitive to badly formed bounding boxes. Top-down is also more sensitive to occlusion as it makes it difficult for the human detector to detect all humans and produce a suitable bounding box. Occlusion is also a problem for the single person pose estimator since there might be multiple occurrences of the same key-point type which would be confusing for a network trained on images containing only one person. The inference time for a pipeline with top-down approach increases with the number of people because a single person pose estimation is estimated for every person. Since the bottom-up approach detects all key-points directly the execution time of detecting key-points is not dependent on the number of present humans, however the execution time of connecting joints to a full pose is affected [12].

OpenPose [13] is an open sourced real-time multi-person pose estimation network. The network uses a bottom-up approach to estimate the poses of multiple persons. Poses are ensemble using part affinity fields a 2D direction vector which is estimated for all detected joints. The complete pose is then created using the direction vector to find connected joints. As other parts of the pipeline require object detection this work will use a top-down approach with a single person pose estimator called High Resolution Net [14](HRNET). HRNET is a single person pose architecture network which predicts poses using the heat-map approach described earlier. HRNET can be used as a multiple person pose estimator by combining the pose estimator with a human detector, which will work in a top-down approach. HRNET performs state of the art precision on the COCO-dataset for single person pose estimation. The reason of the high precision is that the network preserves high resolution throughout the network. In most single person pose estimation network the input is processed through series of high-to-low modules which extracts features and down samples the resolution. To produce the heat-map the resolution is upsampled to the desired output. HRNET connects the high-to-low modules in parallel where for each stage of down sampling a high-to-low module have information from earlier resolution and its own down sampled resolution. As the resolution is preserved throughout the process there is no need for upsampling to the desired output size which create more precise heat-maps.

2.4.3 Human Mesh Recovery

Human Mesh Recovery [15] (HMR) is a neural network which estimates 3D joints and a mesh which is parameterized by shape parameters and 3D joints angles. The model is built using two modules. An encoder to extract features from the image and a regression module to estimate the parameters from features extracted from the encoder. The architecture of the model is created in a way which allows for training using images with only ground truth labels of 2D joints. This is favorable since the number of in-the-wild images with 2D joints labels are much larger than the amount of images with corresponding ground truth 3D joints or mesh. The model is able to train on images with annotated 2D joints by defining the loss as the projection loss between the estimated 3D joints and the annotated 2D joints. Using only the projection loss creates no incitement for the model to produce valid human meshes, only using the projection loss might result in meshes with intersecting body parts or unnatural joint angles. This problem is addressed using a network which classifies the estimated parameters as a valid parameter configuration. Such network is often referred to as a discriminator network. The output of the discriminator is combined with the projection loss which enforce the model to generate meshes which the discriminator classifies as real. The estimated parameters are used to generate a human mesh by using Skinned Multi-Person Linear [16] (SMPL) model. SMPL is an encoding of the human body to represent a wide range of shapes and poses. The model is taught by training on data of 3D meshes in different poses and shapes to reduce the number of parameters needed, to express the mesh. To represent a human body in the SMPL model 82 parameters are needed 10 parameters ι to express shape and 72 parameters β to express the rotations of the joints. The mesh is generated by a linear function with the parameters as input. The created mesh consist of 6890 vertices and 23 joints. Since HMR use projection loss it also needs to estimate the camera model to be able to project the generated 3D joints to the image plane. The camera is modeled using a rotation matrix $R \in \mathbb{R}^{3x3}$ a translation vector $t \in \mathbb{R}^2$ and a scale factor $s \in \mathbb{R}$. Thus, the estimated parameters θ is $\{\iota, \beta, R, s, t\}$. An iterative regression module is used which estimates the parameters iteratively rather than estimating them directly. The regression module uses the extracted features from an image and the parameters from the previous iteration to estimate the update of the parameters, $\nabla\theta$, this is repeated for a fixed amount of steps to estimate the parameters. At the first iteration the parameters are set to the zero which is iteratively updated to the final estimation. The iterative process can be expressed as

$$\theta_{i+1} = \theta_i + \nabla\theta_i. \tag{2.10}$$

Chapter 3
Method

3.1 Data

The data used in this thesis consists of three video recordings from Swedish soccer games. The recordings are recorded in resolution of 4k(3840x2160). The video is recorded using a fixed camera where the translation and rotation is unchanged during the whole recording. The cameras are mounted in such a way that one half of the soccer pitch is visible, an example frame from the videos is shown in figure 3.1. There is a radial distortion in the videos since a wide lens camera have been used for the recordings. Using the algorithm presented in [17] where the distortion factor is calculated from estimating a physically straight line as an arc in the image plane, at least three lines are needed to obtain a good result. A line is annotated by annotating points along that line. The distortion factor is used to rectify the image i.e. removing the lens distortion. When the image is rectified a pinhole camera model and a homography can be estimated from an image. The pinhole camera model is estimated by first estimate the intrinsic parameters. In this work only the focal length is estimated, the skew is assumed to be zero and the principal point is assumed to be in the center of the image. The focal length is estimated using two vanishing points [17]. Using the estimated intrinsic parameters, the extrinsic parameters, pose and rotation, can be estimated. This problem is often referred to as perspective-n-point and can be solved by 3 points of correspondence [18], however to estimate two vanishing points at least four points are needed. The estimation of the camera matrix is refined by iteratively minimizing the reprojection error between the 3D points and the points in the image plane. The homography requires 4 points of correspondence to be estimated. Calibration is done by annotating correspondent points between the pitch model and the image plane where the pitch model is defined as the coordinates in the physical pitch which position is restricted by the rules of FIFA [19]. Points which are used for calibration is shown in figure 3.2. Note that the dimension of the pitch can vary between venues, however, if the venue is known, points which are dependent of the dimension can be used for calibration. Since the points have common lines, points for calibration can be chosen in such way that they will produce at least two distinct vanishing points.

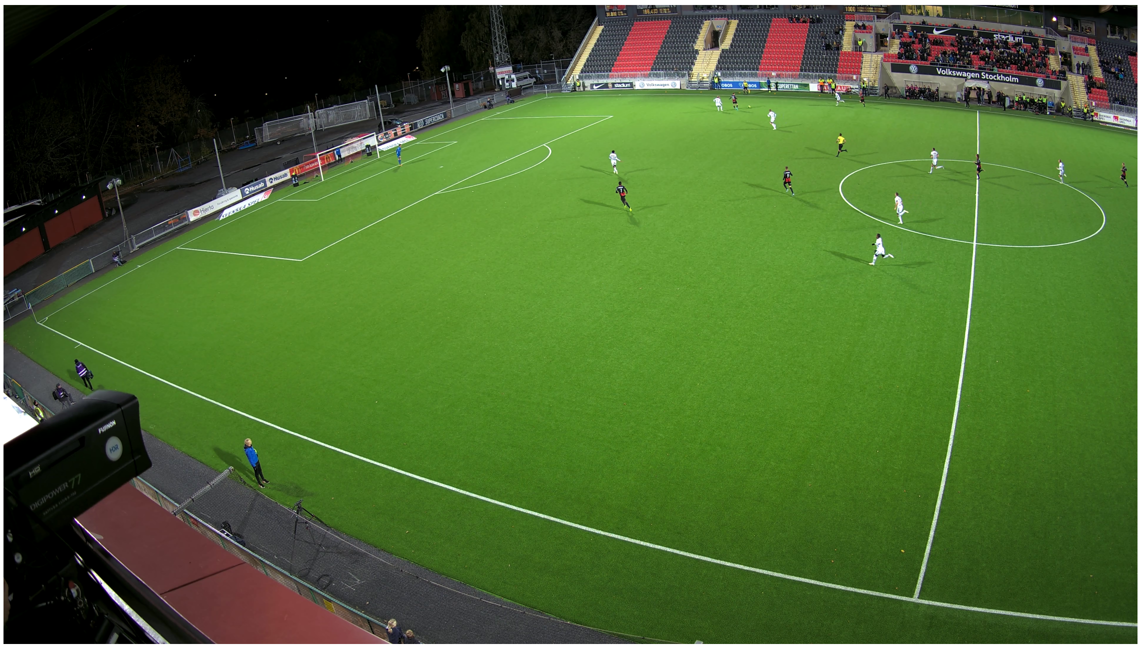


Figure 3.1: An example image from a recorded soccer game

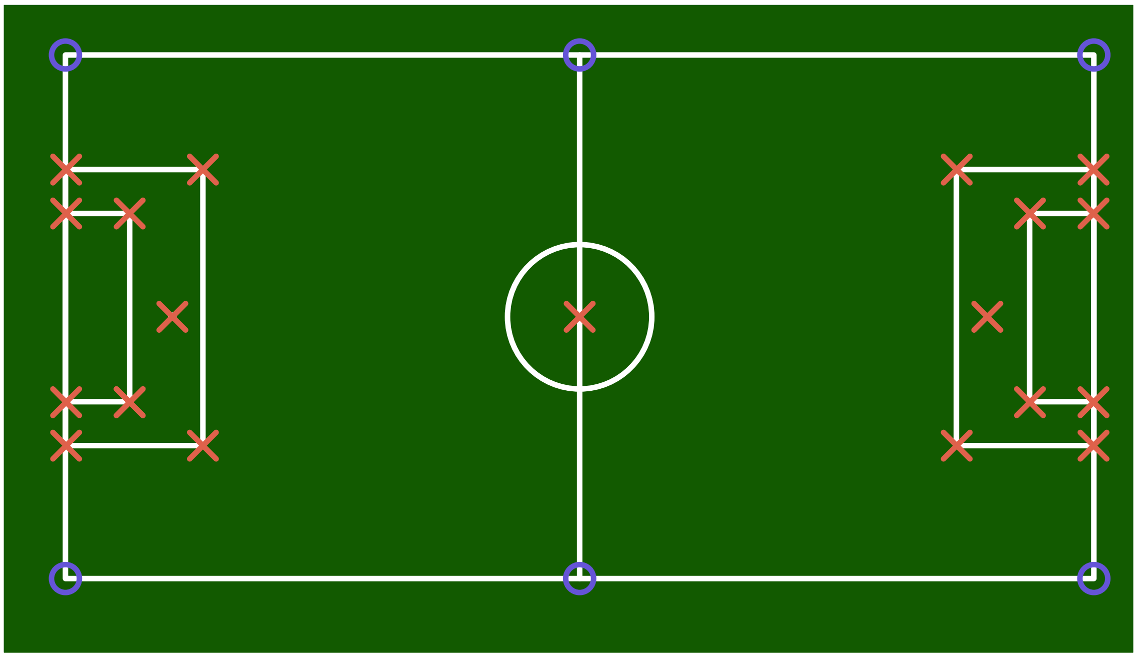


Figure 3.2: Figure shows points used for estimation of the camera model. Red crosses denotes points which are restricted by the rules of FIFA, blue circles denotes points which is defined by the dimension of the pitch.

The video data is sampled to produce a sequence of images per recorded game. Each image will be annotated with direction of the attacking team. An object detector is used to find the players in an image for all images in a sequence. For each player the team identity of the player is annotated and also if the player is in offside position.

3.2 Implementation

The application is implemented using Python [20] and the main libraries that are used are Tensorflow [21] and OpenCV [22].

3.2.1 Player Detection

YOLO is used to detect players and their corresponding bounding boxes. Since the project mainly use Tensorflow, a Tensorflow implementation [23] of YOLOv3 is used. The weights used for the network are the original weights which are used in the YOLOv3 paper. The weights are trained using COCO (Common Objects in Context) [24] data set where humans are represented. To speed up development and since humans are represented in COCO the simplest solution was to use the original weights. It would be time consuming to develop a data set with ground truth detections of soccer players. COCO data set has 80 different classes which YOLO is trained to detect. In this application the only class of interest is human, this is fixed by removing detected bounding boxes that is not of the human class.

The input of the network is an image of size 416x416 which is 1.5% of the pixels in the 4k video data, a down sampling from 4k to 416x416 would result in a large loss of information. To reduce the amount of information loss, each video frame is split vertically into three segments with an overlap between adjacent segments. In this implementation the overlap is 200 pixels. The overlap should be chosen in such a way that the overlap is larger than the width of the desired detections, if the overlap is chosen in this manner there is no risk of splitting an object between two segments. The detected bounding boxes will have a coordinate system where the origin is in the upper left corner of its corresponding segment. To transform the bounding boxes to a common coordinate system the bounding boxes from a segment is translated by the offset of the segment in the original video frame. When all bounding boxes are in a common coordinate system Soft-NMS is used to reduce redundant bounding boxes in the segments and also eventual redundant bounding boxes which might occur in the overlap between segments. Since the camera model and the dimension of the pitch is known the pitch can be projected to the image plane and using the projection detected players can be filtered by removing detections which are not inside the projected pitch or does not intersect with the projected pitch, which can be seen in figure 3.3. In this manner the detection is improved by removing detections of persons which are not currently participating in the game.

Team identity of the players is classified using the histogram model. The histogram models are created using the annotation of team identity in the first frame. For each detected player the shirt of the player is extracted using the pose estimation by creating a bounding box using the joints; left hip, right hip, left shoulder and right shoulder. The technique used to extract the shirts is illustrated in figure 3.4. A histogram model is created for the four different classes; attacking team, defending team, goalkeeper, referee. For all frames except the initial frame the shirt of all detected players is classified using the created histogram model by comparing the intersection between a shirt of a player and the created histogram models. Since there is only one goalkeeper visible from the camera the algorithm classifies at most one player as goalkeeper.

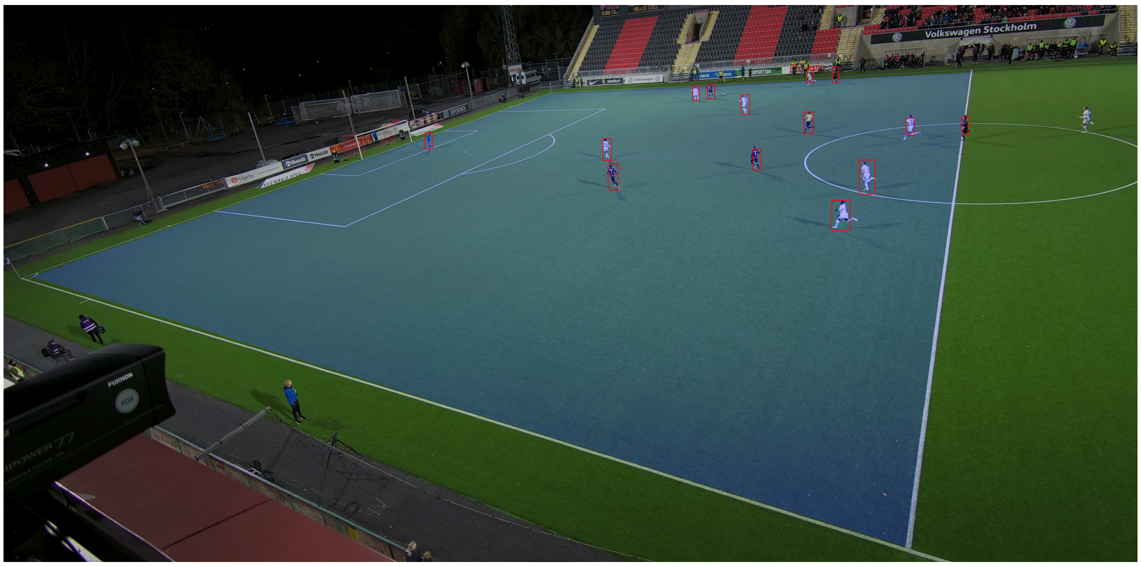


Figure 3.3: The method used to filter detections if their bounding box is outside of the pitch.

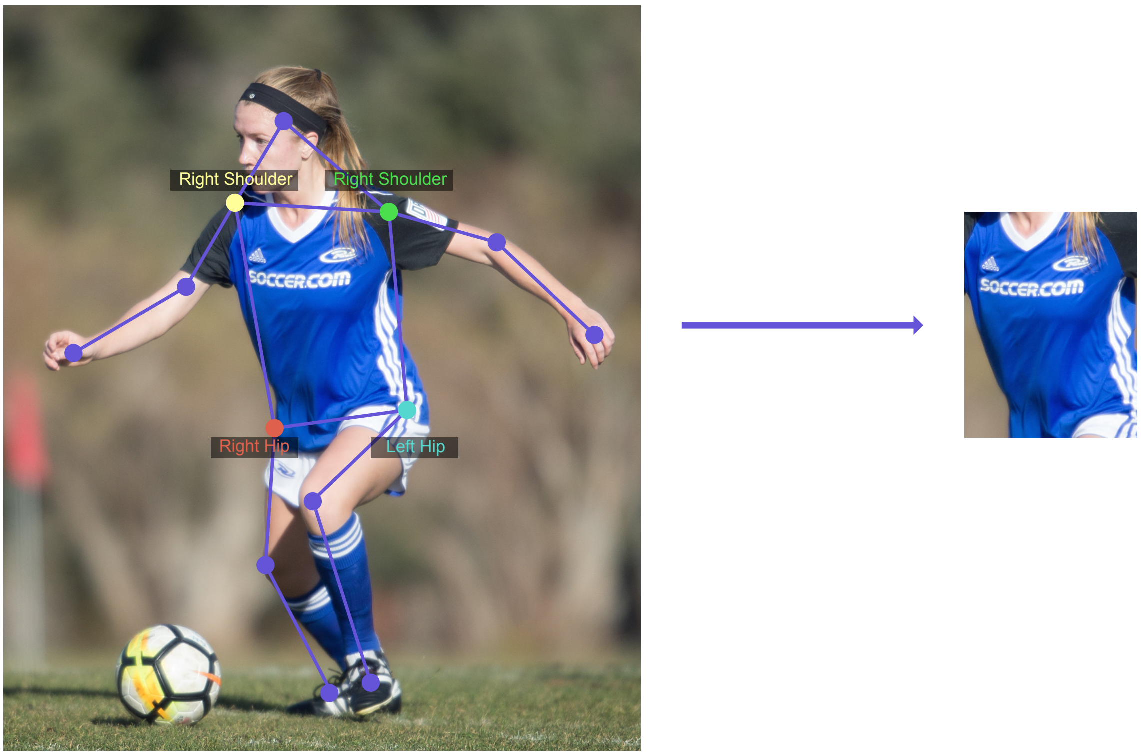


Figure 3.4: Illustration of the method used to extract the pixels of the shirt

3.2.2 Pose Estimation

Human pose estimation for the players will be done using a top-down approach. Since other parts of the pipeline requires object detection of players it is more suitable to use a top-down approach. Furthermore, the loss of information is generally smaller if the cropped image of a player is down sampled to the input size of a single person pose estimation network rather than down sampling the entire frame to the input size of a pose estimation network which uses a bottom-up approach. The implementation will use HRNET single person pose estimation to produce human poses.

The original frame is cropped using the bounding boxes of the detected players in the frame resulting in one image per player. Every player image is resized using letterbox resizing to the desired input size of HRNET which is 288x384. Letterbox resizing is a resizing technique used to preserve the ratio of the image, the resizing is done by resizing both axis equally using the smallest resize factor where the resize factor is $\frac{target\ size}{current\ size}$. The resized images are fed to the HRNET network in one batch resulting in 17 heat-maps for every image. The heat-maps are of size 72x96 and there is one heat-map for every joint. The joints are found by finding the coordinate of the pixel in the heat-map with the largest confidence score. To transform the local coordinate of a joint to the image coordinate system the coordinate is scaled to the input size and then translated by the position of its corresponding bounding box. The result is 17 joints for each player and the corresponding confidence score of the estimated joint. The estimated position of the ankle joints is used to filter out detection of persons which are not on the pitch. This is done by estimating the position on the pitch using the homography.

3.2.3 3D Reconstruction

Generating 3D meshes and 3D key points is done in a procedure similar to the estimation of 2D key points. An image is created for each of the detected players and is used as input to HMR to generate a SMPL model. HMR is used with an input size of 224x224 and the output is estimated by using three iteration loops. The SMPL model provides 3D joints and mesh for a player; however this will be in a local coordinate system for each individual player. The model has no notion of the global world coordinate system and how the different players relate to each other. This problem is solved by matching the earlier estimated 2D joints with the estimated 3D joints and minimizing the projection error. The estimated 3D joints from HMR is in a local coordinate system for each player and the goal is to find the players positions in a global coordinate system. By estimating the players global coordinates, the relation between the players position can obtained. The 3D joints are projected to the image plane using the camera model which have been estimated earlier. The minimization problem is expressed as

$$\min_{\theta,t_x,t_y} = \frac{1}{N}\sum_{i=1}^{N} || \begin{pmatrix} J^i_{2D_x} \\ J^i_{2D_y} \\ 1 \end{pmatrix} - C(R_z(\theta) + \begin{pmatrix} t_x \\ t_y \\ -min(J_{3D_z}) \\ 0 \end{pmatrix}) \begin{pmatrix} J^i_{3D_x} \\ J^i_{3D_y} \\ J^i_{3D_z} \\ 1 \end{pmatrix} ||, \tag{3.1}$$

where J_{2D} and J_{3D} is the estimated joints in the image plane respectively estimated homogeneous 3D joints in a local coordinate system. J^i denotes the index of a joint, the average projection of all joints is used as objective function. R_z is the 4x4 rotation matrix in z axis where θ is the angle of rotation around z axis. C is the estimated camera matrix using the pinhole camera model. Translation t_x, t_y is the translation in x and y axis and $min(joints^z_{3D})$ is the z coordinate of the joint with the smallest z coordinate. By translating the joints by the smallest z coordinate each player is enforced to have at least one joint in contact with the ground. Rotation θ is estimated to find the direction of the player. This is needed since global rotation of z is not estimated by HMR. Translation t is estimated to find the global position of the player. To speed up the convergence the initial guess of t is calculated by using the homography to find how the position of the ankles in the image plane corresponds to its position in the pitch model, the global coordinate system. The rotation θ is initially set to π. The python library SciPy [25] is used to minimize the objective function. SciPy uses Broyden–Fletcher–Goldfarb–Shanno algorithm to search for the parameters which minimizes the objective function. The estimated position of the detected humans can be used to filter the amount of detections by removing humans whose position is outside the pitch.

3.2.4 Detecting Offside Position

When all detected players have been classified to a team and a 3D reconstruction is in place, the matter of classifying if a player is in offside position is a trivial task. The offside rule states that an attacking player is in offside position if it has a playable body part closer to the short side than the defending player with the second closest playable body to the short side, the rule is visualized in figure 3.6. Playable body parts are all body parts except from the arms, the playable body parts can be seen in figure 3.5.The vertices from SMPL which is part of a playable body part is manually annotated. An attacking player is classified as being in offside position if any of its vertices have a distance closer to the short side than any of the vertices of the second closest defender.

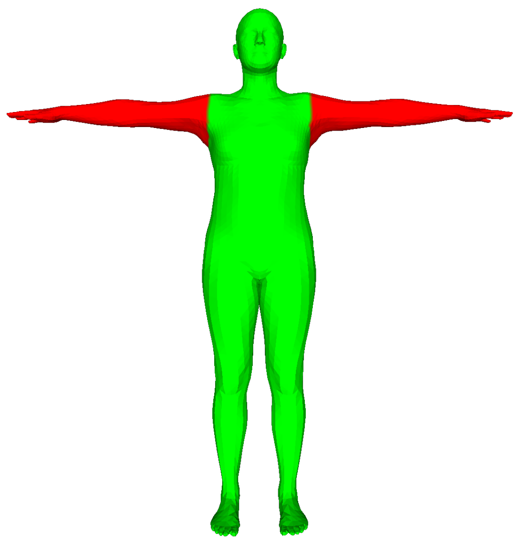


Figure 3.5: SMPL model where playable vertices are green and non-playable body parts are red.

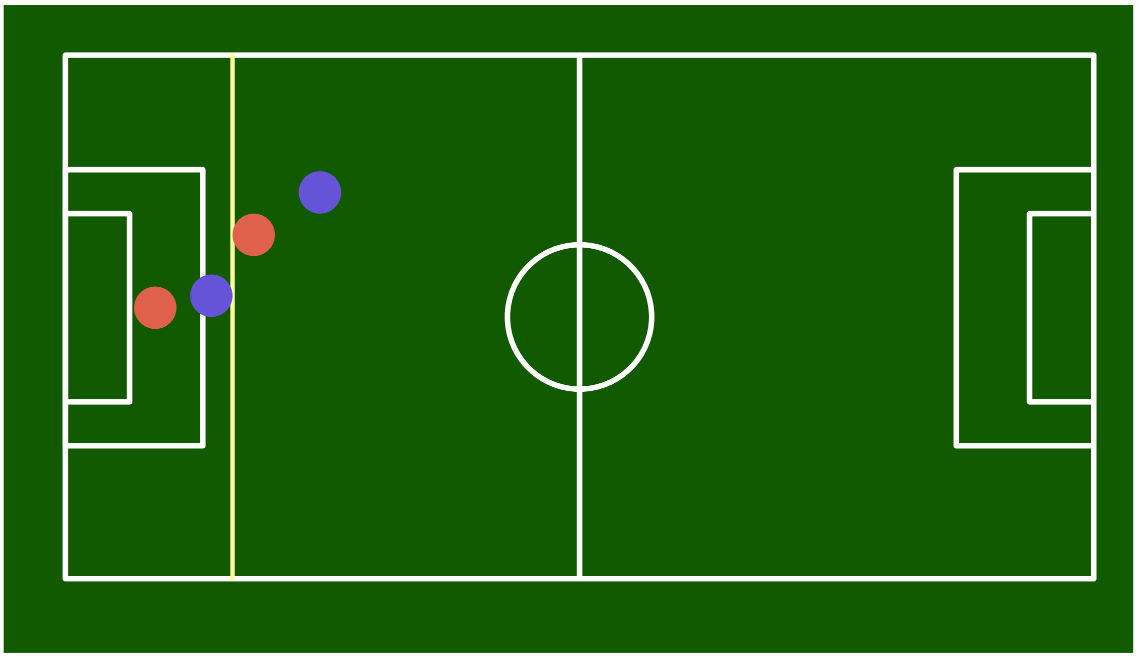


Figure 3.6: Visualization of the offside rule, red circles are defenders, blue circles are attackers and the yellow line represents the offside line

3.2.5 Evaluation

To evaluate the performance of the proposed pipeline it is evaluated with regard to human pose estimation, mesh generation, team classification and offside position classification. Since the resolution of a player varies with the distance between the player and the camera, the effect of the resolution is also evaluated. The resolution might have an effect of the performance of the pose estimation and the mesh generation and therefore these two processes will be evaluated. The difference in resolution will be simulated by down sampling images with ground truth labels to evaluate how the error relates to the resolution. The pose estimation network is evaluated using ground truth data from the COCO data set where the measure of error is object keypoint similarity (OKS) [26]. OKS is a Gaussian function used to evaluate the quality of the pose estimation, it is defined as

$$oks = \frac{1}{N}\sum_{i}^{N} e^{-\frac{d_i^2}{2s^2k_i^2}}, \tag{3.2}$$

where d_i is the Euclidean distance in pixels between the predicted joint and the ground truth, s is the area of the image and k is a predefined factor per joint which controls the fall-off of the Gaussian function. To evaluate the mesh estimation a similar approach is used where images is down sampled. A data set called Unite the People [27] is used as it provides images with ground truth meshes. The data set have been developed by automatically estimating meshes from images of humans and letting human annotators annotate the quality of the generated meshes to filter badly estimated meshes. Evaluation is done using the average euclidean distance between the ground truth vertices and the predicted vertices. The performance of the team classifier and offside position classifier are evaluated with similar metric. Using the annotated, data a classifier is simply evaluated by comparing the ground truth to the prediction.

The performance will be evaluated using the metrics; recall, precision and accuracy. All the metrics depend on the measurement of True Positives, False Positives, True Negatives and False Negatives.

- True Positive(TP) for a class c is the number of correctly classified samples as c
- False Positive(FP) for a class c is the number of incorrectly samples classified as c
- True Negative(TN) for a class c is the number of samples correctly classified as another class than c
- False Negative(FN) for a class c is the number of samples incorrectly classified as another class than c

Accuracy describes the rate of samples that are classified correctly and is defined as $\frac{TP+TN}{TP+FP+TN+FN}$. Precision describes the rate of samples correctly classified as c given all samples classified as c and is defined as $\frac{TP}{TP+FP}$. Recall describes the rate of samples that was classified as c given all samples that belongs to class c and is defined as $\frac{TP}{TP+FN}$. The evaluation of the team classifier is done with regard to the number of bins used in the histogram model to investigate how the number of the bins affects the performance. Offside position classification will be evaluated using three different methods to retrieve the position of the player, using the homography and two variants of the projection loss. Using homography the position of the player is found by estimating how the ankles in the image corresponds to the plane of the pitch, it is considered as the baseline approach. There is two variants of the minimization approach where the difference is in the loss function, one approach uses the loss function defined in equation(3.1). The other variant use a weighted projection loss, similar to 3.1, where the projection loss is weighted using the confidence score from the 2D pose estimation and is defined as

$$\min_{\theta,t_x,t_y} = \frac{1}{\sum_i c_i} \sum_{i=1}^{N} c_i || \begin{pmatrix} J^i_{2D_x} \\ J^i_{2D_y} \\ 1 \end{pmatrix} - C(R_z(\theta) + \begin{pmatrix} t_x \\ t_y \\ -min(J_{3D_z}) \\ 0 \end{pmatrix}) \begin{pmatrix} J^i_{3D_x} \\ J^i_{3D_y} \\ J^i_{3D_z} \\ 1 \end{pmatrix} ||, \tag{3.3}$$

where c is the confidence score from the 2D pose estimation. The purpose of using the confidence as weight is that keypoints which the 2D pose estimator consider certain will have a larger impact when finding the position of the player. In the same manner a keypoint which the pose estimator is uncertain of, for example a joint which is occluded, will have a smaller impact.

Chapter 4

Results

4.1 Pose Estimation

Figure 4.1 shows that the pose estimator HRNET handles lower resolutions well. The figure shows that the OKS score is affected first at a downsampling factor of 0.4 and that the large drop in OKS first occurs at a downsampling factor of 0.2. The average size of the images used to evaluate HRNET's is 381x267. The average size of the detected players are 109x53 however the average of the smallest quarter of detected players is 71x37. A downsampling of 0.2 the images used for evaluation would result in images with the average size of 76x53.

4.2 Human Mesh Recovery

Figure 4.2 show how the mean square error of predicted vertices increase with the decrease of the downsampling factor. The mean square error is unchanged until the downsampling has decreased to 0.5 where the error starts to increase. At around a downsampling rate of 0.3 the error has doubled from an error of 0.1 to 0.2. The average size of the images used for evaluating mesh recovery is 341x232. A downsampling with a factor of 0.5 would result in an average size of 171x116 similarly the average size would be 102x70 with a downsampling factor of 0.3.

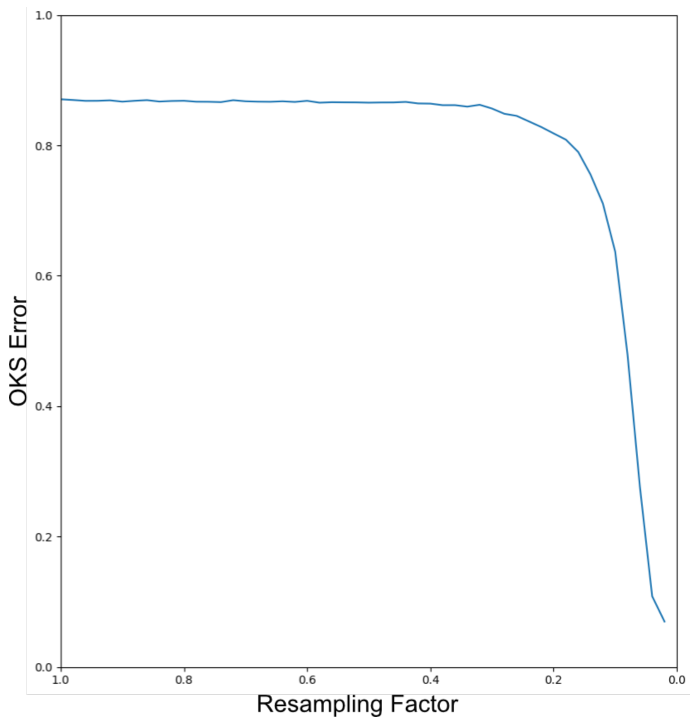


Figure 4.1: OKS score at different downsampling factors. Note that the x axis is flipped, and a large score is good.

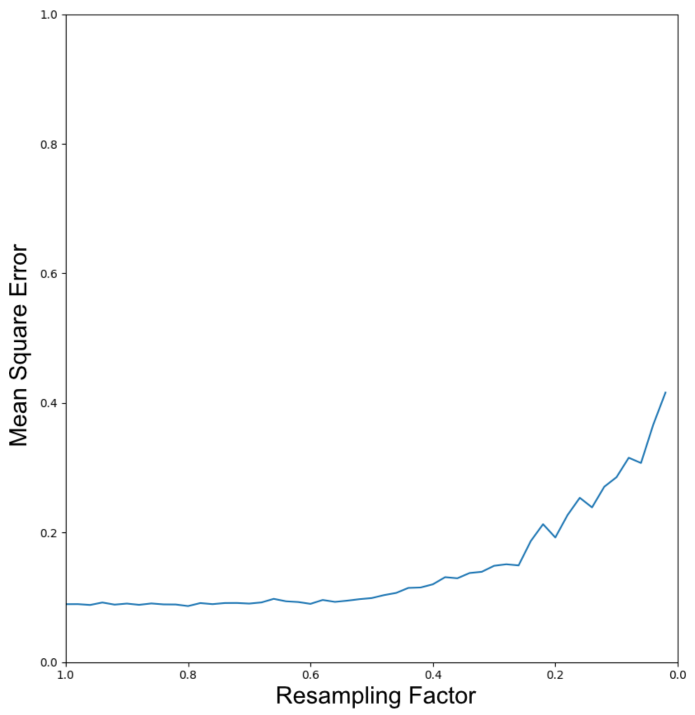
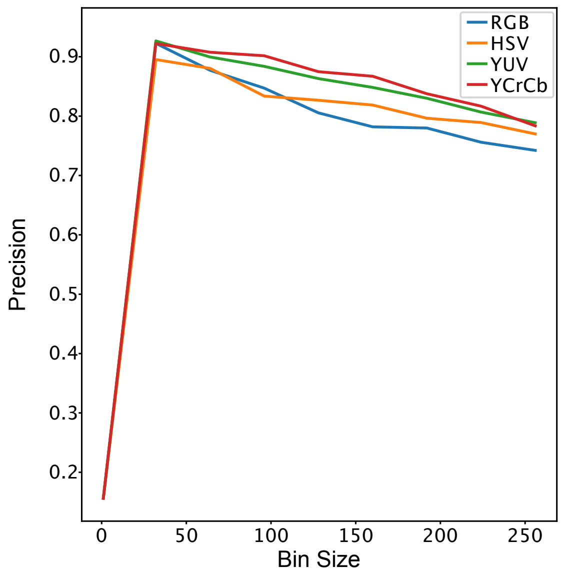


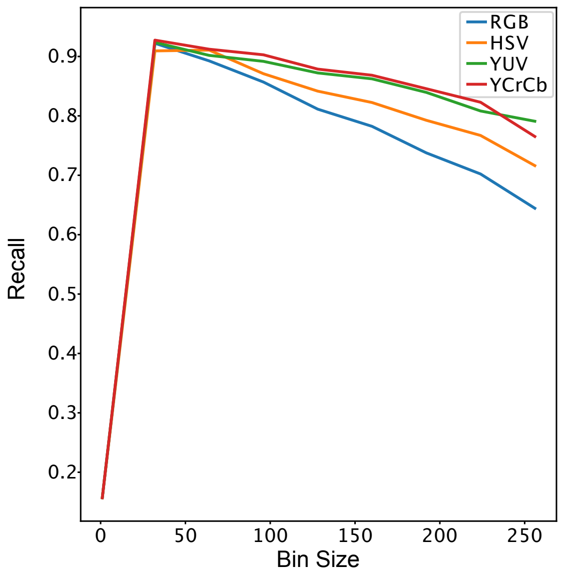
Figure 4.2: Mean square error at different downsampling factors. Note that the x axis is flipped, and a small error is good.

4.3 Team Classification

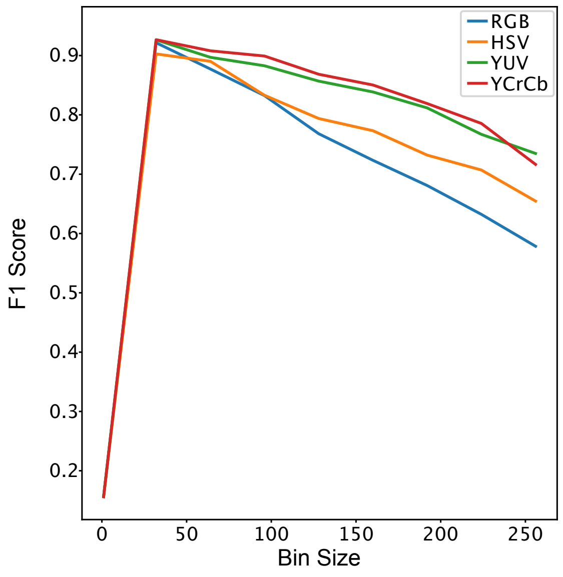
Figure 4.3 shows that the best size of bins is 32 for all color spaces. The color space which performs the best with regard to F1 score is YCbCr although RGB only have a slightly smaller F1 score. Figure 4.4 show the confusion matrix for RGB and YCbCr with a bin size of 32. The precision and recall for RGB with 32 bins is 0.923 respectively 0.909. There is only a few more misclassification with RGB as color space and since the major parts of the systems use RGB as color space it might be preferable to use RGB since that reduces the number of color conversions.



(a) Precision

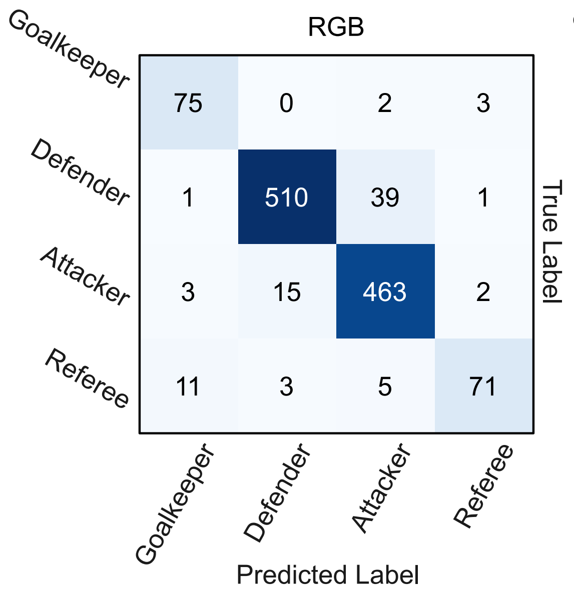


(b) Recall

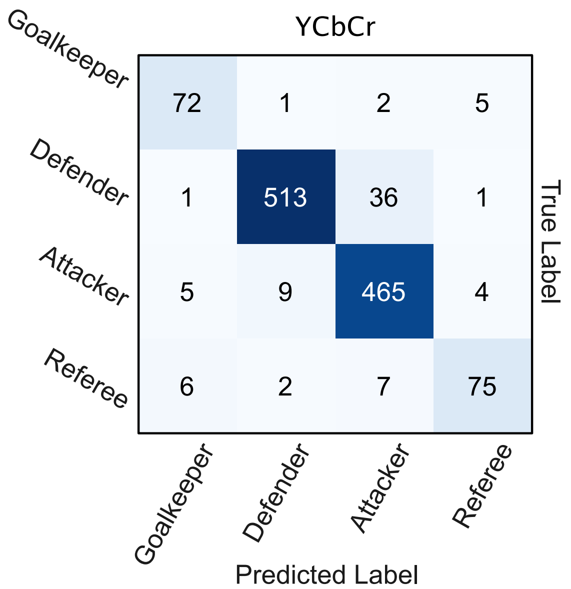


(c) F1 Score

Figure 4.3: Precision, Recall and F1 score of team classification using different color spaces.



(a) RGB with bin size 32

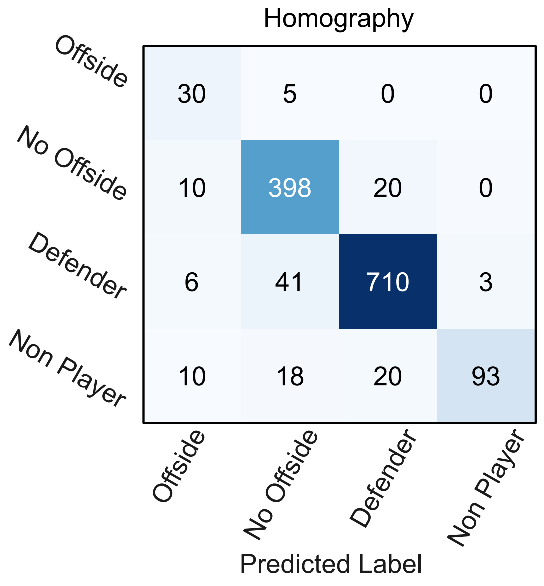


(b) YCbCr with bin size 32

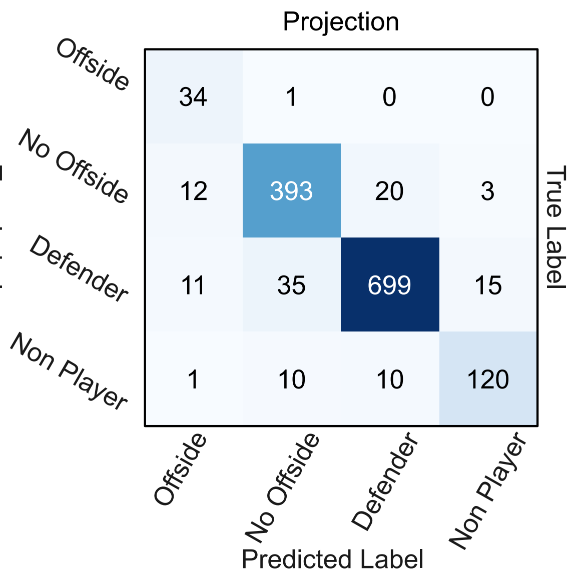
Figure 4.4: Confusion Matrix for the two best performing configurations.

4.4 Offside Classification

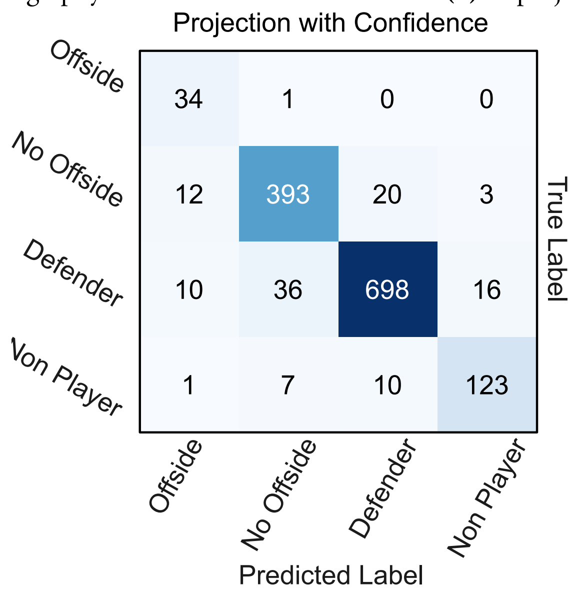
Figure 4.5 and table 4.1 shows that the methods with the best F1 Score are the two methods using 3D reconstruction. Both methods, weighted and unweighted, have the same average F1 Score. This method receives the highest average F1-score and also the highest F1-score for the offside class. Team classifications have been made using a bin size of 32 and the YCbCr color space. The non-player class includes the referee and detections of persons which is not a part of the game such as coaches or substitute players warming up. The detections are filtered to eliminate non-player by checking if bounding boxes intersects with the pitch and using the homography to calculate if a person is on the pitch, however the methods which use the 3D reconstruction detections can further be filtered using the position retrieved from the 3D reconstruction. This explains why the F1 score of the non-player class is higher for the methods using the 3D reconstruction. In figure 4.6 and table 4.2 the true team identity is used which differs from the earlier discussed metrics where the team identity was predicted using the histogram model. The goal is to isolate the offside classification from the team classifier. Figure 4.6 and table 4.2 shows that the best performing method is 3D reconstruction with an unweighted reprojection error, however the metrics for the different methods are very similar. The largest difference is the precision of the offside class between the methods using weighted and unweighted reprojection error. As seen in figure 4.6 the difference between the methods is the number of false positive classifications of non-offside as offside. The two cases where the weighted and unweighted method differs can be seen in figure 4.7.



(a) Homography



(b) Reprojection Error



(c) Weighted Reprojection Error

Figure 4.5: Confusion matrix for offside classification for the different methods of retrieving player position.

Class	Precision	Recall	F1 Score
Offside	0.54	0.86	0.66
No Offside	0.86	0.93	0.89
Defender	0.95	0.93	0.94
Non Player	0.97	0.66	0.78
Average	0.83	0.85	0.82

(a) Homography

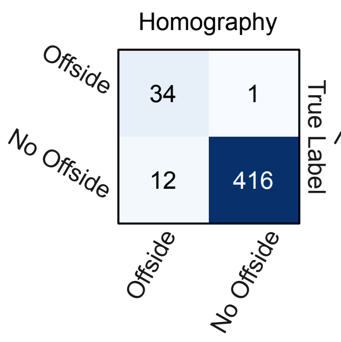
Class	Precision	Recall	F1 Score
Offside	0.59	0.97	0.73
No Offside	0.90	0.92	0.91
Defender	0.96	0.92	0.94
Non Player	0.87	0.85	0.86
Average	0.83	0.92	0.86

(b) Reprojection Error

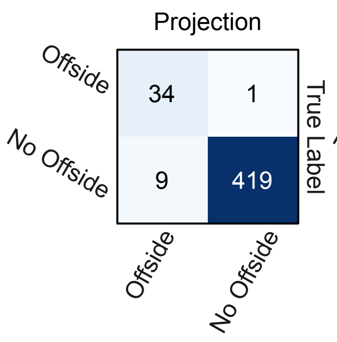
Class	Precision	Recall	F1 Score
Offside	0.60	0.97	0.74
No Offside	0.90	0.92	0.91
Defender	0.96	0.92	0.94
Non Player	0.87	0.87	0.87
Average	0.83	0.92	0.86

(c) Weighted Reprojection Error

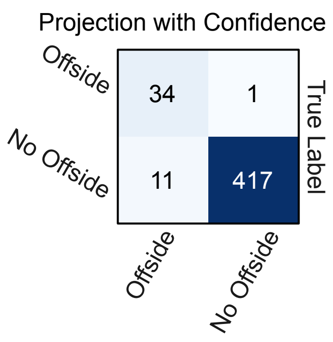
Table 4.1: Precision, Recall and F1 Score for offside classification for the different methods of retrieving player position.



(a) Homography



(b) Reprojection Error



(c) Weighted Reprojection Error

Figure 4.6: Confusion matrix for offside classification using the true team identity of the players.

Class	Precision	Recall	F1 Score
Offside	0.74	0.97	0.87
No Offside	1.00	0.98	0.99
Average	0.87	0.97	0.91

(a) Homography

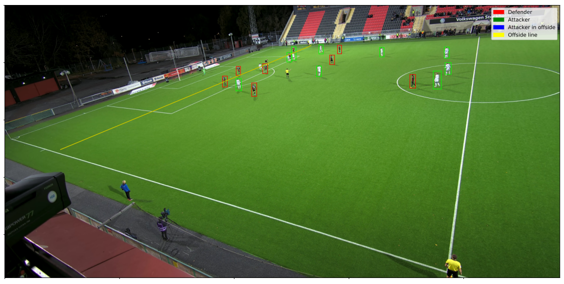
Class	Precision	Recall	F1 Score
Offside	0.79	0.97	0.85
No Offside	1.00	0.98	0.99
Average	0.89	0.98	0.93

(b) Reprojection Error

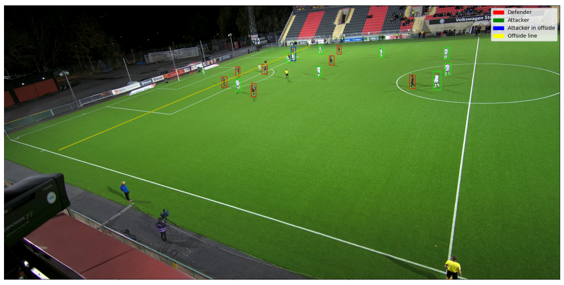
Class	Precision	Recall	F1 Score
Offside	0.76	0.97	0.85
No Offside	1.00	0.97	0.99
Average	0.88	0.97	0.92

(c) Weighted Reprojection Error

Table 4.2: Precision, Recall and F1 Score for offside classification using the true team identity of the players.

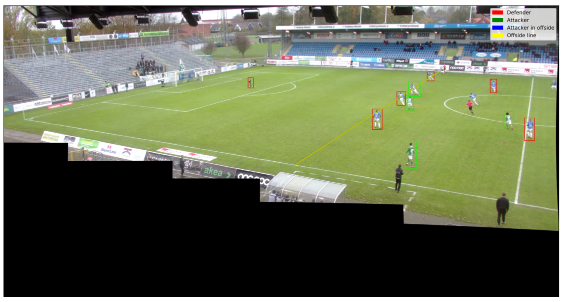


(a1) Unweighted reprojection error

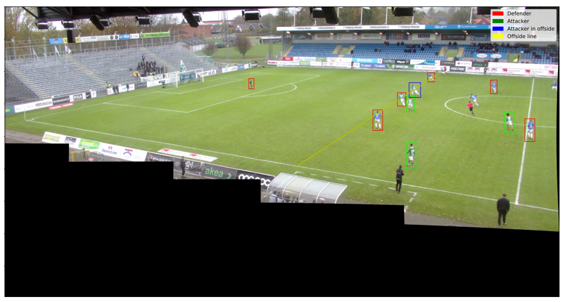


(a2) Weighted reprojection error

(a) Case 1



(b1) Unweighted reprojection error



(b2) Weighted reprojection error

(b) Case 2

Figure 4.7: The two cases where the classification of methods using unweighted and weighted reprojection error differ

Chapter 5 Discussion

It is clear in figure 4.1 and 4.2 that the scale of players affect the accuracy of the predicted meshes and keypoints. It is expected that the performance is affected by the scale as the amount of information varies with the scale, however the threshold where the performance is affected varies between the estimators. With regard to pose estimation the size of a person must be quite small compared to the input size to have an effect on the OKS score. As mentioned earlier the average size of the smallest quarter of detections is 71x37. The OKS score for the pose estimator start to decrease rapidly around the same size which indicates that there are players where the pose estimation is inaccurate because of the distance to the camera which will affect the estimated position of the player. Mesh recovery is more sensitive to the size of the detected player where the error starts to increase at an average size of 171x116 and the error is doubled at a size of 102x70. As the error starts to increase at 171x116 and the average size of the detected players are 109x53 there is probably a large number of players where the quality of the reconstructed mesh is affected negatively due to the distance to the camera and therefore also the accuracy of the estimated position. Since this project tries to solve the classification with only one camera this is not an easily solved problem. If multiple cameras from different positions and views were used the players would hopefully be at least close enough to at least one camera. Another possible solution would be to have a larger resolution of the recorded video or have multiple cameras and stitch together the recorded videos to simulate a larger resolution. On the other hand, the recorded videos are already at a 4k resolution and increasing the resolution further would result in more data and most likely the runtime would be slower.

In figure 4.3 it is shown that a bin size of 32 is preferred for the team classifier to receive the best result, regardless of which color space. As the bin size increases above 32, all metrics decrease resulting in worse performing team classifiers. One possible explanation to this is that when the bin size is large there is a too hard limit between colors which is considered to be equal. As the players move across the pitch the color of the shirt will be affected by the lightning and shadows resulting in colors with different intensity. Using a smaller bin size gives a larger color space of what colors is considered to be equal. It should be noted that there will always be a strict limit between the bins which could cause incorrect classifications.

A solution would be to use some smoothing method to smooth the occurrences of a color to neighboring colors which could have a positive effect on the performance of the team classifier. It is shown that the color space YCbCr is the color space which performs best. YCbCr has a channel called luma which represents the brightness of a color which is suitable for classifying soccer shirts since in most cases one team has a dark shirt and the other team has a bright shirt. The performance of RGB is close to that of YCbCr which might be explained by that there is one recorded game where the teams have green respectively blue shirts where RGB works very well for separating the shirts. Given that there is only three recorded games to evaluate the team classification it would probably require more data to draw any further conclusions.

In table 4.1 it is shown that estimating the players position using 3D reconstruction performs better than using only the homography. As shown the methods using 3D reconstruction are better at detecting non-players which indicates that these methods are more robust at estimating the position of a player, since these methods are better at detecting when a person is outside of the pitch. Detecting non player have an impact on the ability to detect players in offside position which can be seen in table 4.1 the methods with a higher recall of non-players have a higher precision and recall of the offside and no offsides classes. This is reasonable since when a non player is classified as a defender it can potentially change the position of the offside line having the effect of classifying offsides as non-offsides. This can be an explanation why the homography method has more misclassifications of offside as no offside which can be seen in figure 4.5. When comparing tables 4.1 and 4.2 one can see the impact of the team classifier, in table 4.2 where the true team identity is used the metrics are rather high for all three methods which indicates that the team classifier is the bottleneck whilst the different methods for finding the position of the players works well. The misclassifications of team identity might also be an explanation to the low precision of the offside class in table 4.1 several defenders are classified as being in offside position which only is possible if they have been classified as attacking players. As seen in figure 4.6 there is still false positive offside classifications when the true team identity is used. Figure 4.7 shows two cases where reconstruction using weighed reprojection error incorrectly has classified two players as offside. Commonly for both these cases are that the object detector either missed players or that the predicted bounding box is incorrect. In figure 4.7a the bounding box contains two players which confuses the pose estimator since it has been trained on single person images resulting in a bad pose estimation which affect the ability to classify the team of the player as well as the position of the player. Similarly, in figure 4.7b the object detector fails to detect the defender behind the attacking player. The object detector could be retrained using training data of players at a soccer pitch which could increase the performance of the object detector. Retraining YOLO with appropriate training data would probably solve the issue of missed players or predicted bounding boxes with multiple players, however it would not solve the issue in figure 4.7b where the defending player is largely occluded. Occlusion is hard to solve using only one camera since there is very little information about the occluded player available in the image. Using multiple cameras from for example the opposite sides of the pitch would probably solve a large amount of issues related to occlusion, on the other hand it would introduce other problems such as syncing the video between the cameras and merging the reconstruction produced from the different cameras.

One major takeaway from this thesis is that the performance of a system, with components depending on each other, is never better than the worst performing component. Since the predictions from a neural network is used to create predictions in another network the error in predictions propagates through the system resulting in errors in the final classification. Using the proposed technique, it is possible to detect players which is in an offside position with a high recall, however the precision is lower resulting in many false positives. There is clearly more work needed to get better results especially with regard to the team classifier. It should also be clarified that the annotations of offside is annotated using single images, therefore the performance of the system is compared to offside which the human eye can detect from a single image.

Chapter 6
Future Work

As mentioned in the discussion the two components which reduces the system is the object detector and the team classifier. To further develop the system the object detector needs to be improved by retraining the detector with training data which is relevant to the task of identifying soccer players. This was outside the scope of this project since it would require a lot of work and time to annotate training data. The team classifier could be improved by using a more sophisticated method than the histogram model which is rather simple. As shown in the result and mentioned in the discussion the performance of the system is heavily improved when using the true team identity of the players.

The thesis has not considered the inference time of the system and a system used to detect offside position should preferably be able to run in real time. As this has not been taken into account further optimizations of the system are needed to lower the inference time. Optimizations could be done by quantizing the neural networks to save bandwidth and speed up the mathematical operations. There could also be optimizations made by rewriting critical components using a programming language which gives the programmer more control such as C++ rather than Python.

The proposed technique only tries to solve one part of the offside problem. Offside if the receiver of a pass is in offside position when the pass if the ball is passed to a player in offside position. To solve the detection of offside there is several other components needed such as ball detection, detecting a pass and tracking all players. The proposed solution to estimating if a player is in offside position could be used as a part to detect offside, however it requires more work to solve the entire problem. The technique proposed to reconstruct the players on the pitch can also be used in other application apart from offside detection. A possible application can be to simple view the game from a different view than from the camera recordings. The technique for reconstruction can also be used in other fields where a reconstruction of the humans in a scene is usable. Soccer has the advantage of having a pitch which can be represented by a plane and with visual features making it easy to estimate the homography and projection matrix.

Chapter 7 Bibliography

[1] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65.6 (1958), pp. 386–408. ISSN: 0033-295X. DOI: 10.1037/h0042519. URL: http://dx.doi.org/10.1037/h0042519.

[2] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[3] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[4] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. arXiv: 1506.01497 [cs.CV].

[5] T. Lin et al. “Focal Loss for Dense Object Detection”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2999–3007.

[6] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *arXiv* (2018).

[7] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[8] Dongxing Wang. *Information Science and Electronic Engineering: Proceedings of the 3rd International Conference of Electronic Engineering and Information Science (ICEEIS 2016), January 4-5, 2016, Harbin, China*. 1st. USA: CRC Press, Inc., 2016. ISBN: 1138029874.

[9] Navaneeth Bodla et al. “Soft-NMS – Improving Object Detection With One Line of Code”. In: (2017).

[10] Michael J. Swain and Dana H. Ballard. “Color indexing”. en. In: *International Journal of Computer Vision* 7.1 (Nov. 1991), pp. 11–32. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/BF00130487. URL: http://link.springer.com/10.1007/BF00130487 (visited on 04/27/2020).

[11] Q. Dang et al. “Deep learning based 2D human pose estimation: A survey”. In: *Tsinghua Science and Technology* 24.6 (2019), pp. 663–676.

[12] Miaopeng Li et al. “Bottom-up Pose Estimation of Multiple Person with Bounding Box Constraint”. In: *CoRR* abs/1807.09972 (2018). arXiv: 1807.09972. URL: http://arxiv.org/abs/1807.09972.

[13] Zhe Cao et al. *Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. 2016. arXiv: 1611.08050. URL: http://arxiv.org/abs/1611.08050.

[14] Ke Sun et al. *Deep High-Resolution Representation Learning for Human Pose Estimation*. 2019. arXiv: 1902.09212. URL: http://arxiv.org/abs/1902.09212.

[15] Angjoo Kanazawa et al. “End-to-end Recovery of Human Shape and Pose”. In: *Computer Vision and Pattern Regognition (CVPR)*. 2018.

[16] Matthew Loper et al. “SMPL: A Skinned Multi-Person Linear Model”. In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34.6 (Oct. 2015), 248:1–248:16.

[17] Aiqi Wang, Tianshuang Qiu, and L.-T Shao. “A Simple Method of Radial Distortion Correction with Centre of Distortion Estimation”. In: *Journal of Mathematical Imaging and Vision* 35 (Nov. 2009), pp. 165–172. DOI: 10.1007/s10851-009-0162-1.

[18] T. Tamminen and J. Lampinen. “Sequential Monte Carlo for Bayesian Matching of Objects with Occlusions”. In: *IEEE Transactions on Pattern Analysis Machine Intelligence* 25.06 (June 2006), pp. 930–941. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2006.128.

[19] The International Football Association Board. “Laws of the Game 2020/21”. In: (2020).

[20] *Python*. URL: https://www.python.org/ (visited on 06/05/2020).

[21] *Tensorflow*. URL: https://www.tensorflow.org/ (visited on 06/05/2020).

[22] *OpenCv*. URL: https://opencv.org/ (visited on 06/05/2020).

[23] Kevin Costa. *Tensorflow-YOLOv3*. https://github.com/kcosta42/Tensorflow-YOLOv3. 2019.

[24] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[25] *SciPy*. URL: https://www.scipy.org// (visited on 09/02/2020).

[26] Coco Common Objects in Context. *Keypoint Evaluation*. URL: https://cocodataset.org/#keypoints-eval (visited on 06/05/2020).

[27] Christoph Lassner et al. “Unite the People: Closing the Loop Between 3D and 2D Human Representations”. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. July 2017. URL: http://up.is.tuebingen.mpg.de.

Master’s Theses in Mathematical Sciences 2020:E74
ISSN 1404-6342

LUTFMA-3431-2020

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/