Football Height Estimation from a Monocular Camera

TSBB11 Technical Report

Linnea Fridman
linfr737@student.liu.se

Karin Fritz
karfr040@student.liu.se

Angelina Johansson
angjo675@student.liu.se

Annette Lef
annle554@student.liu.se

Victoria Nordberg
vicno213@student.liu.se

Lukas Tegendal
lukte246@student.liu.se

Supervisor: Abdelrahman Eldesokey
abdelrahman.eldesokey@liu.se

ISY - The department of Electrical Engineering

December 20, 2018

Abstract

This project aims to determine if it is possible to estimate the height of a football from the ground using a monocular camera. This will be tested with two different approaches, an analytic solution using geometry and a non analytic solution using neural networks. To be able to test the methods, two types of data were generated. Coordinates of trajectories were simulated and projected to an image or synthetic images were generated using a game engine. Experiments shows that given the ball size in the input, both analytical and non-analytical solutions performs well, but when noise is added to the data the performance is reduced. When the ball size is not available, a recurrent neural network gives good performance, even with noisy data. When using synthetic images with a convolutional neural network, the results were not satisfactorily, probably due to the small size of the dataset.

1. Introduction

This project is a part of a project course at Linköping University and the project task is given by an external company, the customer. The customer develops software used to provide real-time sport analytics for sport teams. Their software allows users to track the ball during a game in real-time. This makes it possible to analyze a lot of data, for example how much time the ball spends on different sides of the field and how many passes are made between players. For the football example, the image data is collected from two cameras placed at the same longitudinal side of the field. Each camera takes footage of one half of the field, and the output from the two cameras forms a complete image of the whole field. A drawback with the current method used by the customer is that more parameters than the image coordinates has to be known to be able to determine the 3D position of the ball. In the current method the ball height is always set to zero even when the ball is in the air, which creates a faulty image. The project task is to produce an estimation of the height with the same camera setup as the one currently used and preferably without using the size of the ball as a parameter.

2. Problem formulation

The project aims to determine the 3D position of a football in real-time using a 2D image sequence from monocular cameras. Image coordinates of the ball are assumed to be known. The problem that remains and has to be solved is determining the height of the ball relative to the ground using only 2D images and the corresponding image coordinates of the ball. The height above the ground of the ball should be estimated by two different measures. One as a numerical value and another as a classification with different intervals, for example on the ground, reachable or unreachable. The size of the ball can be used, but the final goal is to estimate the ball's height above the ground without using its size.

The real data contains noise and therefore the model should be robust to noisy data. There exists no annotated real data, instead synthetic data will be created and used.

3. System overview

The system has two main components: data generation module and ball height estimation module. The input to the data generation module is a camera matrix and the output is either trajectories consisting of sampled ball coordinates or images with ball coordinates generated with a gaming engine. The trajectories consists of 3D coordinates, the size of the ball and the corresponding 2D image coordinates.

The ball's height above the ground is calculated by an analytic method or estimated by models using neural networks. Both solutions are implemented in Python. The models using neural networks will be developed with Keras and consist of both regression and classification models. To speed up the training of the networks Google colaboratory was used [1]. Another tool used to simplify the work with the networks was to use Jupyter [2]. See figure 1 for an overview of the system.

4. Creating synthetic data

Since no annotated real data is available, the first task is to create synthetic data. The first type of data is created by projecting coordinates from 3D trajectories following parabolic curves. The generated data will consist of 2D coordinates, the ball size and the height in 3D coordinates. The second data type is a set of images portraying a football field with players, goals, a football etc. This data should also contain both 3D world coordinates and 2D image coordinates of the football in the images.

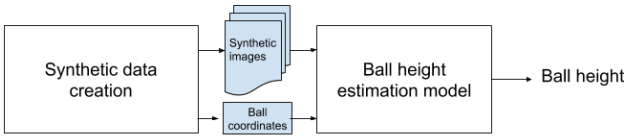

Figure 1. System overview.

4.1. Coordinates from a trajectory

To create the 3D trajectory, a start position, x_0, a velocity, v_0 in x-, y-, and z-direction and a sampling rate f_s are defined. The time of flight of the ball is calculated with equation 1 where v_z is the velocity in z-direction and g is the gravity. For each sample i, during the time of flight, the 3D position of the ball is calculated with equation 2.

$$t = \frac{2v_z}{g} \tag{1}$$

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \begin{pmatrix} v_{0_x} \\ v_{0_y} \\ v_{0_z} \end{pmatrix} t_i - \begin{pmatrix} 0 \\ 0 \\ 0.5g \end{pmatrix} t_i^2 \tag{2}$$

Multiple trajectories are created by randomizing the start position and velocity. When making multiple trajectories at the same time, the start position is chosen to be the end position of the previous trajectory. This is to make a long coherent sequence of trajectories.

To transform the 3D coordinates into 2D image coordinates, a camera matrix is used to calculate the position of the ball in the image plane, see equation 3 where $(u, v, 1)^T$ are the homogeneous image coordinates, $(x, y, z, 1)^T$ are the homogeneous world coordinates K are the internal camera parameters, R is the rotation of the camera and t is the translation of the camera. The radius of the ball in the image is calculated by projecting the center of the ball and a point on

the edge of the ball into the image and calculating the distance between those two points.

$$\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = K \begin{bmatrix} Rt \end{bmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{3}$$

Examples of trajectories are shown in Figure 2. To make the synthetic data resemble real data, noise is added to 30% of the data. This value was selected through a visual investigation and should simulate the resolution uncertainty of the camera. When including ball size in the model, the noise is added to ball size. The noise is uniform in the range $[0, 0.5max(s))$, where $max(s)$ is the maximum ball size in image coordinates, from all points in the data set. When the ball size is not included, the noise is instead added to the ball position. This noise is uniform in the range $[0, 0.05max(x))$ and $[0, 0.05max(y))$. The noise is set as uniform because in the real data the noise will be in images and a quantization error can be compared to noise with a uniform distribution. Data with noise added to ball position is shown in Figure 3.

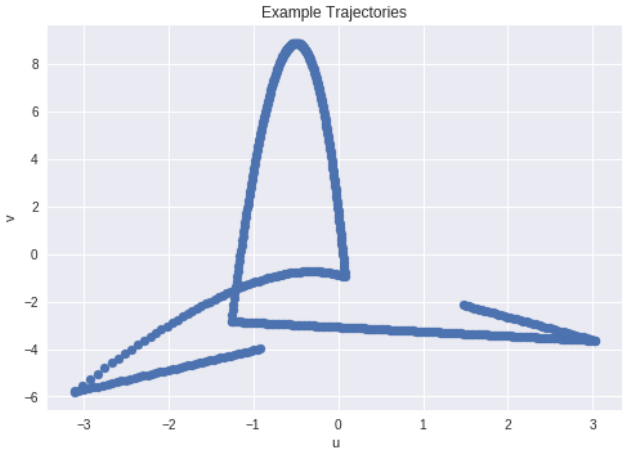

Figure 2. Trajectories without added noise to the coordinates.

To simulate the ball being passed between players on the ground, 50% of all trajectories follows the ground instead of moving through the air.

4.2. Images

For the second set of synthetic data, images are created in the game engine Unity[3]. A world is created to resemble a football field with players and a football.

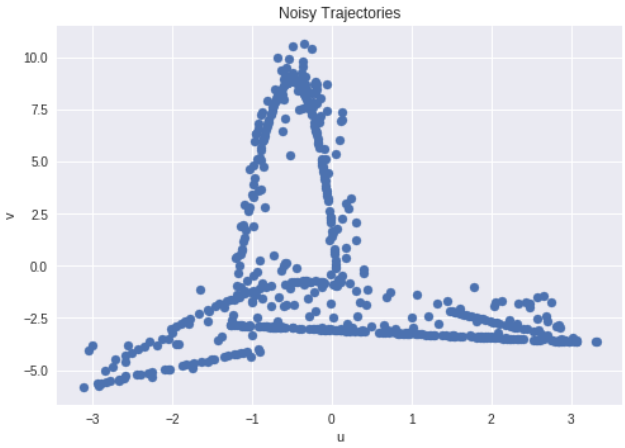

Figure 3. Trajectories with added noise to the coordinates.

The players are programmed to move around on the field to make the scene look as realistic as possible. The ball is programmed to move in a direction specified by inputs from the keyboard. The keyboard input "Space" will make the ball "jump" and move along a parabolic trajectory. The height of this jump is programmed to be around 3 meters. See Figure 4 for a sample of the image data set.

Figure 4. Sample from the image data set.

To generate image data with corresponding football coordinates, a script in Unity is created to take screenshots and save the football's coordinates while the game is running. To achieve this, a game camera given in Unity is used and the football's world coordinates are saved and stored in a file with a specific ID depending on when the screenshot was taken. Corresponding screenshot is saved with the same ID and using these coordinates and the camera matrix, the football's position in image coordinates can be retrieved by using equation 3.

5. Solutions

The problem was solved with two different approaches and three different types of inputs. The ap-

proaches was either analytic or non-analytic and the different inputs were either coordinates with ball size, coordinates without ball size or image data.

The non-analytic solutions are different types of neural networks. When training the networks the data is divided so that 80% is used as training data and 20% is used as validation data. For the trajectory data, both the training data and validation data are normalized according to equation 4 where the mean and variances are calculated on the training data.

$$data = \frac{data - mean}{variance} \tag{4}$$

Different parameters such as number of nodes, number of layers, types of loss functions etc. are tested to see which network would give the highest accuracy on the validation data. All networks are trained until convergence.

The classifier models are evaluated based on the final accuracy and loss on the validation data and the regression models are evaluated based on the final mean square error on the validation data. The results of the classifiers are visualized by plotting both the ground truth and the predictions for some of the projected points from the validation data. The result of the regression are visualized by plotting the height of the ball for some of the points in the validation data, this is done for both ground truth and predictions.

5.1. Solutions using the ball size

Initially, the problem is solved using the size of the ball as part of the input.

5.1.1 Analytic solution

The analytic approach of determining the 3D position x of the football is constructed by knowing the radius of the ball r, the camera matrix C and the image coordinates of the center and the edge of the ball y_c and y_r. The geometry of the problem can be seen in Figure 5. The solution can be found by following the lines l_c and l_r, which have directions defined by equation 5, where y_i is a point in the image and c_c is the camera center. The 3D position of the football (in a camera centered coordinate system) is found by following these lines until the distance between them are equal to the radius of the football. To transform these coordinates from camera coordinates to world coordinates, the position and orientation of the camera are used.

$$l_i = y_i - c_c \tag{5}$$

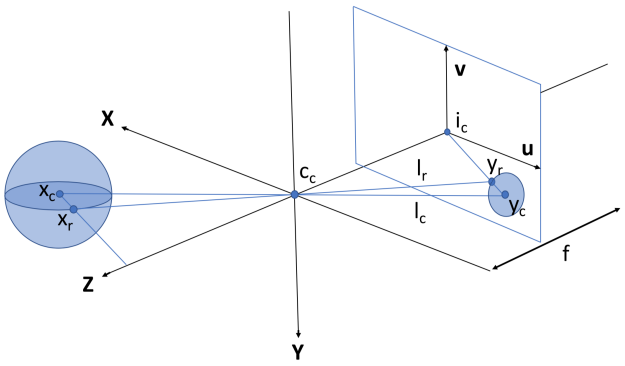

Figure 5. Geometric model used for the analytic solution.

5.1.2 Fully connected networks using coordinates from a trajectory

Another way to solve the problem is using a non-analytical approach. Initially, a simple fully connected network is created and trained with the coordinate vectors retrieved from the trajectories. The inputs to the network are the x-position, the y-position and the size of the ball in the image. The ground truth is given by the z-position, i.e. the height of the ball.

Both regression networks and classifiers are constructed since the height estimation is supposed to be retrieved as a numerical value as well as one of the following classes; on ground, reachable and unreachable.

5.2. Solutions without using the ball size

Since the problem is based on deciding the height of the football from an image during a real-time football game, the size of the ball in the image can be very noisy. Especially the size will be very small and therefore sensitive to noise. This gives us reason to find solutions that would work without the use of the football's size.

5.2.1 Fully connected networks using coordinates from a trajectory

The fully connected networks are also tested without the use of the ball size. This to see if it is possible to

achieve a good result even with this simple network architecture. The ball size is removed from the previous fully connected networks. The training data and input only consists of the ball position in image coordinates.

5.2.2 Networks using time dependencies

It is not possible to calculate the ball height analytically without knowing the ball size in image coordinates. Using the camera model described in Section 5.1.1, the ball might be positioned anywhere on the line l_c. An alternative approach is to use temporal information from previous image frames to provide an estimate of the height. In addition to the ball position in the current frame, inputs from the n previous images are used to estimate the ball height. The hypothesis is that the movement pattern will provide enough information to correctly classify the input. The data used was the ball positions in image coordinates as training data and a binary classification of the ball height (on the ground or in the air) as ground truth. Two methods for performing binary classification were implemented, one using a fully connected network with multiple coordinates as input, and one using a recurrent neural network (RNN) with either LSTM or GRU modules [4]. Both models use sequences of multiple points as input. To classify the ball height at time t, input $x_t, x_{t-1}, ..., x_{t-n-1}$ are used, where x_t is the ball position in image coordinates at time t and n is the number of time steps used as input. Given a long continuous sequence, shorter sequences of length n are sampled randomly. This allows for a large number of training data to be generated from a sequence of limited length.

The basic idea behind recurrent neural networks are that they use the fact that data is dependent on previous data [4]. RNN's are built up of cells passing the temporal data through the network. These cells can contain different transfer functions and can be connected into different architectures. One such architecture is Long-Short-Term-Memory (LSTM) which takes both long-term and short-term dependencies into account. The LSTM networks are here trained to classify whether the ball is on the ground (within some interval) or in the air. If multiple consecutive LSTM layers are used, the LSTM modules can return sequences instead of singular values. Another transfer function is gated recurrent unit (GRU) [4] and it can handle dependencies in the data at different time scales.

5.2.3 Networks with image input

Using images as input to the network can provide some additional information that will make it possible to determine the height of the ball without the size parameter. Such information could be relations between ball height and the height of football players. When using images as inputs, the networks used are convolutional neural networks (CNN). A CNN is a neural network that has one or more convolutional layers, followed by one or more fully connected layers. CNN's are often used for image inputs. They are also easier to train since they have fewer parameters than a fully connected network with the same number of hidden units. [5]

To make it possible to combine both images and the extracted x and y image coordinate of the ball as inputs, a fourth dimension of the image is added. This dimension consists of a heat map with a Gaussian bell that has its maximum centered around the position of the football. This adds a feature to the network providing information about the position of the ball.

Both a regression type of network and a classifier will be constructed and evaluated with CNN. The actual height of the football is of higher interest when using image data, the classifier will have a larger amount of classes than the other classifiers. This is done to create a classifier that in some way can be resembled to a regression type of network. The continuous values are more interesting when using images as input. The additional information provided by the images should allow for a more accurate estimates, not possible using only the ball position.

6. Result

The following section will present the result of the different methods.

6.1. Solutions using the ball size

The results given in this subsection are produced by methods using the ball size as a parameter.

6.1.1 Analytic solution

The analytic solution is able to solve the problem and give an exact value of the 3D position of the ball as long as the camera is oriented in such way that it is ”looking“ at the ball.

6.1.2 Fully connected networks for classification using coordinates from a trajectory

A simple fully connected network that worked well for the classification problem was constructed as in Figure 6. Fc stands for fully connected and the number beside indicates the number of neurons in the layer. The input to the network is the x-coordinate, y-coordinate and ball size in the image and the output is the probabilities of which height class it belongs to. The loss that was used was categorical cross entropy and the used optimizer was Adam with a batch size of 128. The final accuracy after 30 epochs on the validation data was 99%.

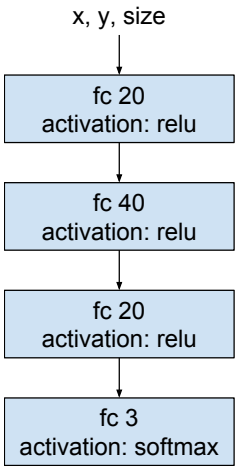

Figure 6. Architecture of the simple fully connected network for classification.

The ground truth and predictions from the model are plotted for some of the points. The points in the plots are the 3D world points of some of the trajectories projected in an image. These can be seen in Figure 7 and 8. The different colors of the points indicates which class it belongs to, green is ground, blue is reachable and grey is non-reachable. As can be seen, the points that are misclassified are the ones that are on the border between classes.

Different depths and sizes of the layers were also

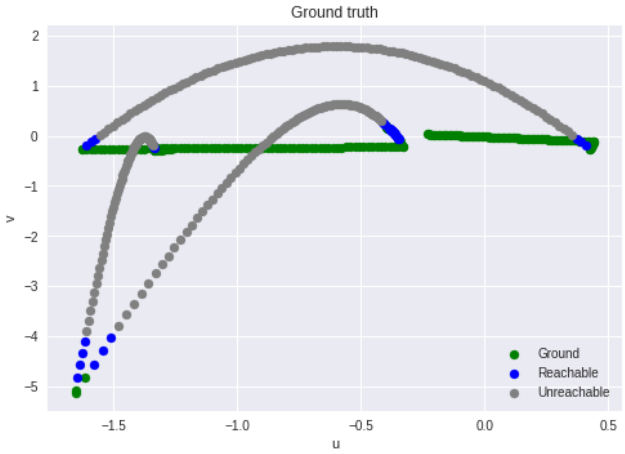

Figure 7. Ground truth for the simple fully connected network for classification.

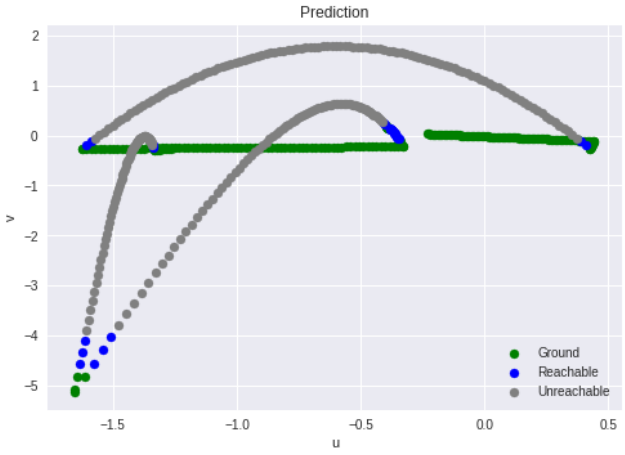

Figure 8. Prediction for the simple fully connected network for classification.

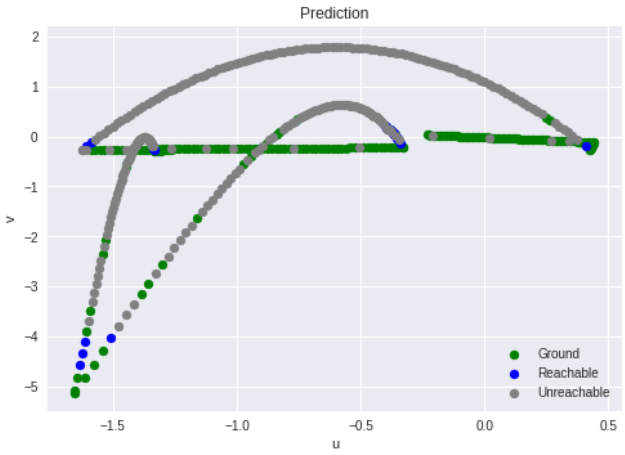

Figure 9. Prediction for the simple fully connected network for classification with noisy ball size.

tested. A smaller network gave lower accuracy. A larger network gave a slight increase in accuracy. The result when using different activation functions were approximately the same. When using stochastic gradient descent as optimizer the accuracy gets 84%. Changing the batch size and learning rate had low impact on the result.

Since the result from the classifier was good, noise was added to the ball size and the same network architecture was tested again. The prediction for some of the points are plotted in Figure 9, which should be compared to the ground truth in Figure 7. The prediction for noisy data has worse result than the network that trained without noisy data.

6.1.3 Fully connected networks for regression using coordinates from a trajectory

A network architecture that worked well for the regression problem can be seen in Figure 10. As for the classification problem the input to the network is the x-coordinate, y-coordinate and ball size in the image. The output is however the ball height, i.e. the z-coordinate. The loss that was used was mean squared error and optimizer Adam with batch size of 128. The final mean squared error on the validation data was 0.11. The ground truth and predictions, i.e. the heights, from the model are plotted for some of the points, see Figure 11.

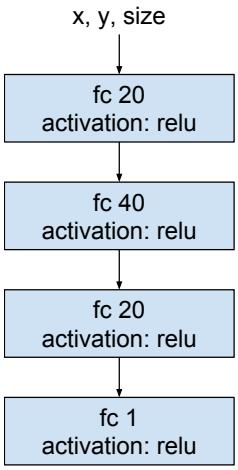

Figure 10. Architecture of the simple fully connected network for regression.

As for the classification network, different depths and sizes of the layers were also tested. Both a smaller and a larger network gave higher mean squared error. The result when using different activation functions, learning rates and batch sizes were approximately the same.

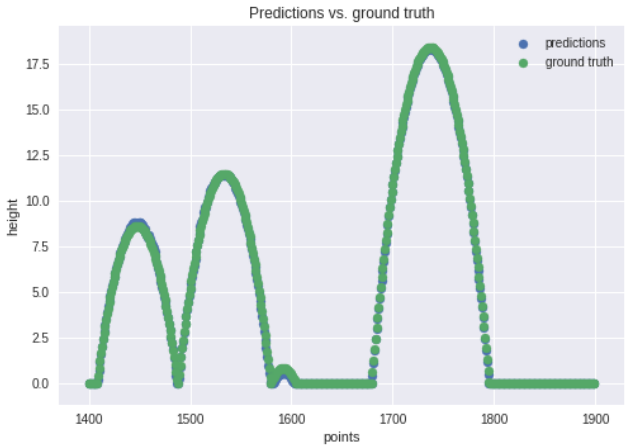

Figure 11. Prediction vs. ground truth for the simple fully connected network for regression.

As for the classification network noise were added to the ball size after finding an architecture that worked well. Predictions and ground truth from the model are plotted for some of the points in Figure 12. As can be seen, the predicted points follow the curve of ground truth, but are not able to determine when the ball is on ground since the points are most of the time predicted to be above height zero.

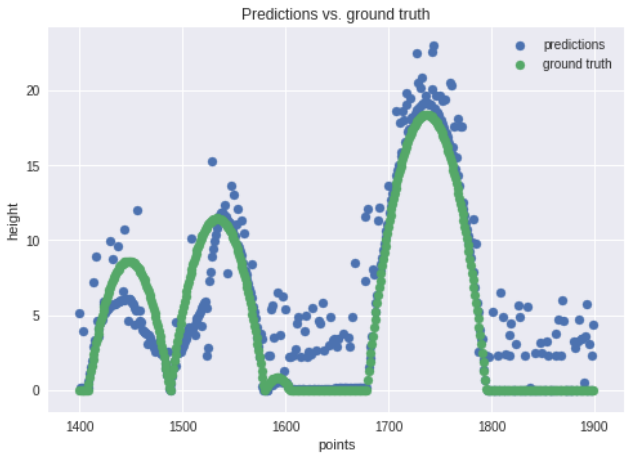

Figure 12. Prediction vs. ground truth for the simple fully connected network for regression with noisy ball size.

6.2. Solutions without using the ball size

The results given in this subsection are produced by methods which are not using the ball size as a parameter.

6.2.1 Fully connected networks for classification using coordinates from a trajectory

The same network architecture that was used with the size of the ball was tested when removing the ball size from the input to see if the network could solve the problem without the ball size. The prediction of some of the points can be seen in Figure 13, which should be compared to the ground truth in Figure 14. When not including the ball size, no points are classified as reachable and a lot as ground.

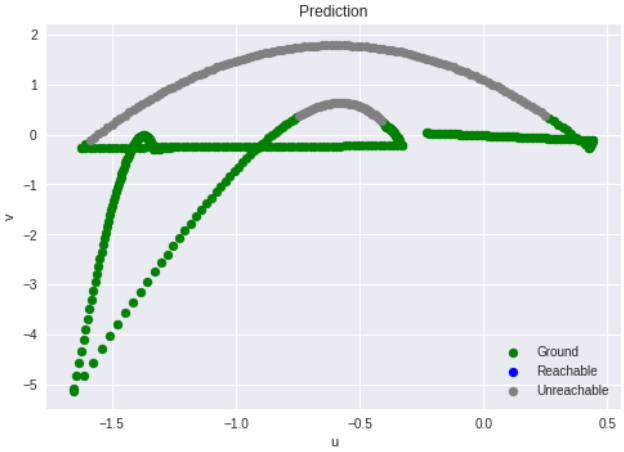

Figure 13. Prediction for the simple fully connected network for classification without ball size.

6.2.2 Fully connected networks for regression using coordinates from a trajectory

The ball size was also removed from the input of the fully connected regression network to see if the network could solve the problem without using the ball size. The final mean squared error on the validation data was 22.9 and the ground truth and predictions for some of the points can be seen in Figure 15. The model is able to follow some of the trajectories while some is more close to the ground than the actual trajectory.

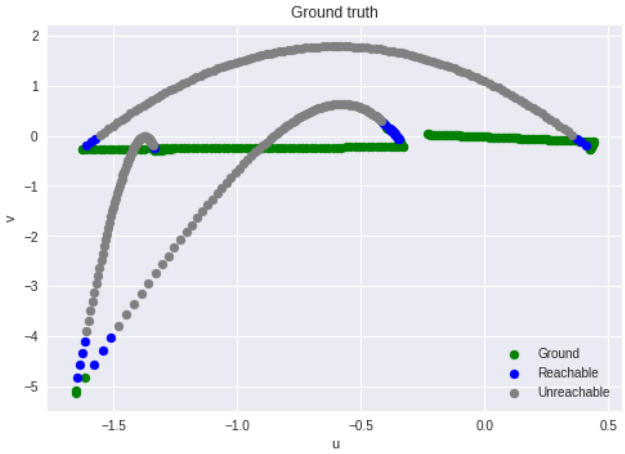

Figure 14. Ground truth for the simple fully connected network for classification.

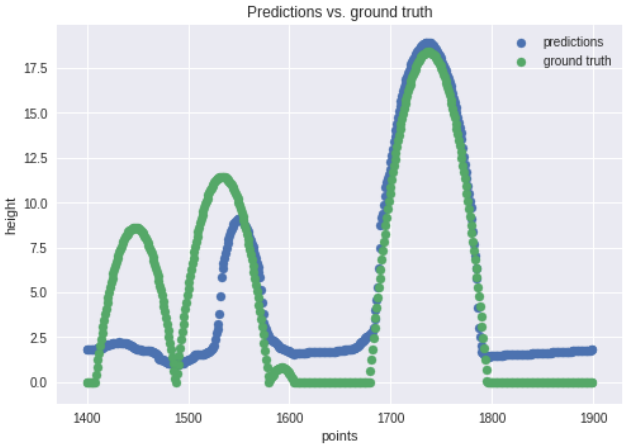

Figure 15. Prediction vs. ground truth for the simple fully connected network for regression without ball size.

6.2.3 Networks using time dependencies

Given the same training data, different LSTM network architectures and using different optimizers gave similar results. In following section, the result from one of the networks will be described but the same result was achieved with other architectures. Also the result from one network built with GRU layers will be presented.

Using LSTM layers the network could perform classification satisfactorily with some problems with low trajectories. Low trajectories were sometimes classified as ”ground” even though the height really was in the ”air” interval.

The layers of the network for where the performance will be shown can be seen in Figure 16. The first and second layer are LSTM layers with 32 nodes,

and have ”return_sequences” set to true. The third layer is also a LSTM layer with 32 nodes but with ”return_sequences” set to false. Lastly, the fourth layer is a dense layer with one output.

The input to the LSTM was an array of 15 coordinates and the output was an array containing zeros and ones representing the classes ”ground” and ”air”. The batch size used was 100, the optimizer was rmsprop and the loss function was binary_crossentropy. 150000 training sequences were sampled from a dataset of 10000 trajectories. After 50 epochs the accuracy was 95%. Figure 17 and 18 shows a prediction and ground truth for a part of the validation data.

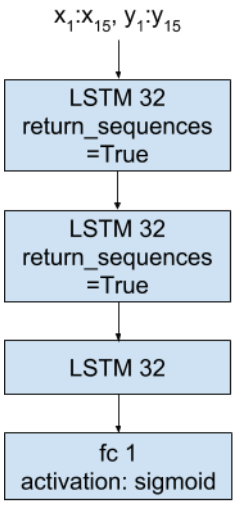

Figure 16. Architecture of the LSTM network.

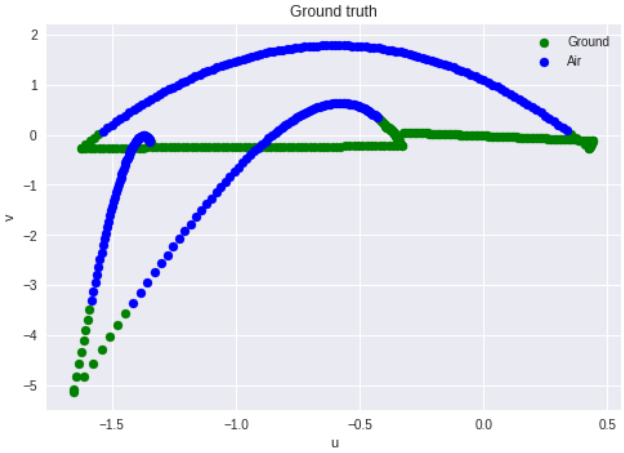

Figure 17. Ground truth for the LSTM network and GRU network.

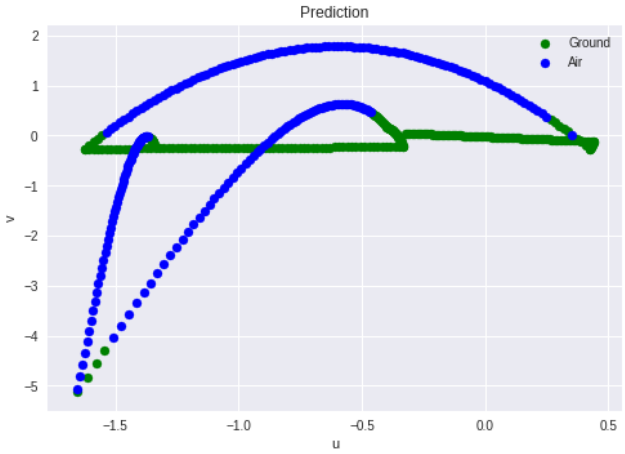

Figure 18. Prediction made by the LSTM network.

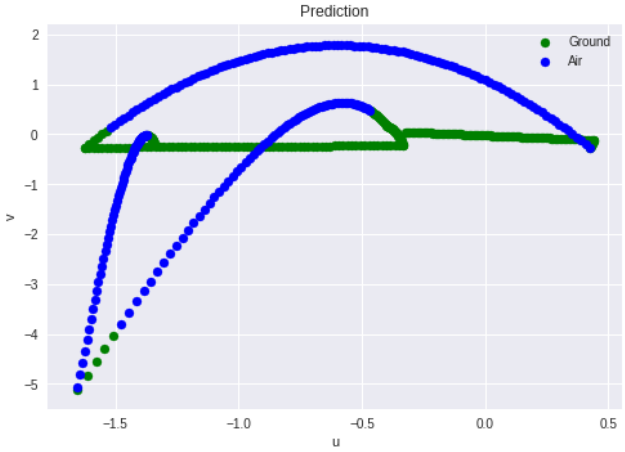

Figure 19. Prediction made by the GRU network.

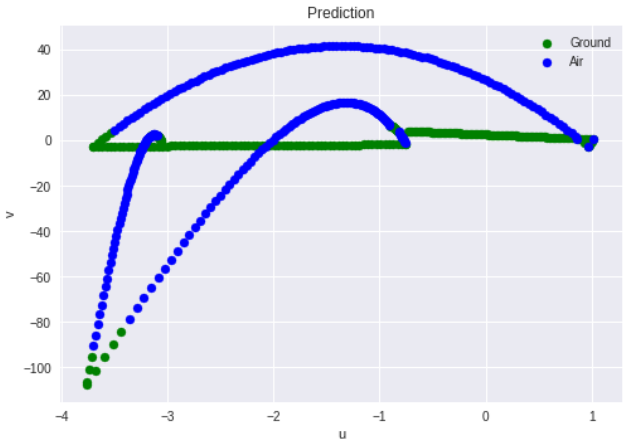

Figure 20. Predicion made by the LSTM network on noisy data.

The Figures 17 and 19 show the prediction and ground truth made with a RNN using GRU. The net-

Size	Training samples	Validation samples	Training sequence	Validation sequences
100%	689043	174511	40000	10000
10%	69804	17451	40000	10000

Table 1. Datasets sizes used for evaluation

Dataset	Accuracy Fully connected	Accuracy LSTM
100%, without noise	93.62%	94.25%
10%, without noise	94.26%	93.23%
100%, with noise	88.84%	92.4%
10%, with noise	88.6%	91.35%

Table 2. Results for multiple point input models.

work has the same architecture as the LSTM network except that the LSTM layers are exchanged to GRU layers. After 50 epochs the accuracy was 96%.

Figure 20 shows predicted trajectories using the LSTM network on noisy data. The accuracy was 93% after 50 epochs.

LSTM networks were also trained using input data in arrays containing one coordinate (instead of 15). This gave a converging accuracy of 80%. The plots corresponding to the ones in Figures 17 and 18 but for this input showed that the network classified the whole trajectories as air.

To evaluate how dependent the methods are to the amount of training data, the networks with multiple inputs were tested with different amounts of training data. The same amount of sequences were sampled from the different sized datasets. The dataset sizes used are shown in Table 1. The full dataset (100%) corresponds to 10000 trajectories.

Two different models were evaluated. One with 3 fully connected layers, batch normalization and 20% dropout, and one model with 3 LSTM layers. The results are shown in Table 2. When the dataset does not contain noise, both models show similar performance. When adding noise, the LSTM model performs better than the one with fully connected layers, with only a small loss in accuracy compared to the noise free dataset. Using the smaller dataset only has a marginal effect on the results. The accuracy is for the LSTM network is slightly lower than the results obtained during the qualitative analysis. This is likely because less training sequences were sampled from the dataset.

6.2.4 Networks with image input

Different CNN architectures were tested to create a network that uses the 4D images to solve the regression problem. The ground truth and predictions for some of the points can be seen in Figure 21 and 22. These figures shows result using the same network but for different intervals of the validation data. Figure 21 shows the result for sample 1 to 100 and Figure 22 shows the result for sample 200 to 300. The network architecture is shown in Figure 25 where the output is the estimated height. The chosen optimizer and loss function are the same as for the fully connected network described in Section 6.1.3, that is Adam and mean square error. The final mean square error on the validation data, using this network was 0.94.

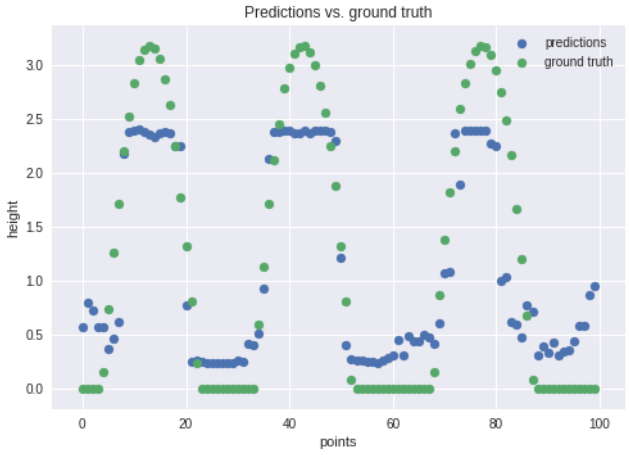

Figure 21. Prediction made by a regression CNN for sample 1 to 100.

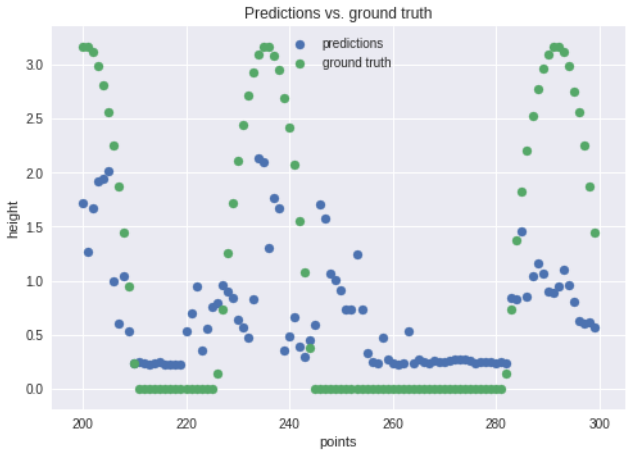

Figure 22. Prediction made by a regression CNN for sample 200 to 300.

The ground truth and prediction for the classification problem can be seen in Figure 23 and Figure 24. The network layers are represented in Figure 26 and the output is given as a probability vector with six classes of height intervals. The loss that was used was categorical cross entropy and Adam was used as optimizer together with learning rate 0.0001 and batch size 20. The final accuracy after 40 epochs was 40%.

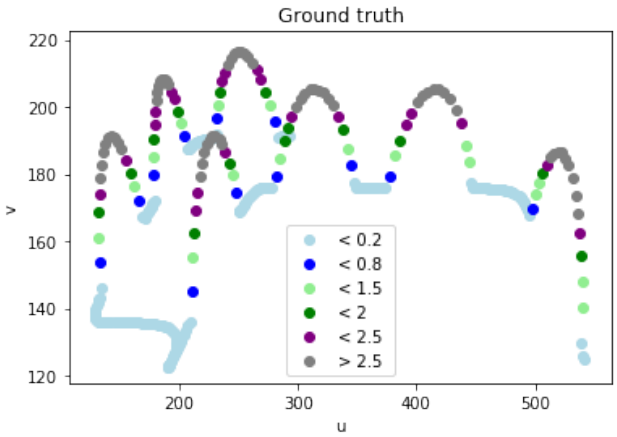

Figure 23. Ground truth for the classification CNN.

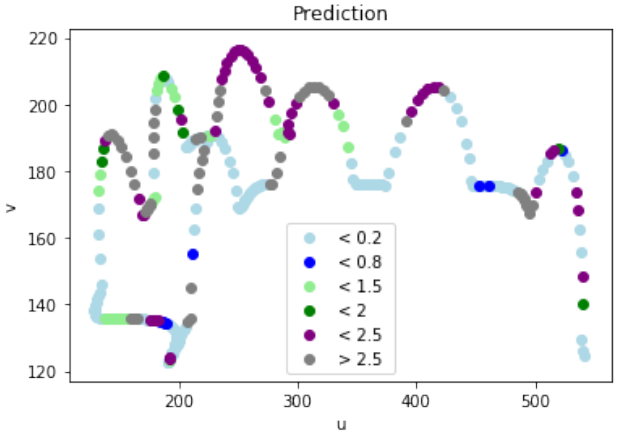

Figure 24. Prediction made by a classification CNN.

7. Discussion

This section will present a discussion about the result.

7.1. Solutions using the ball size

The result when the problem is solved using the size of the ball as a part of the input is good. However, both methods have a hard time handling noise in the input.

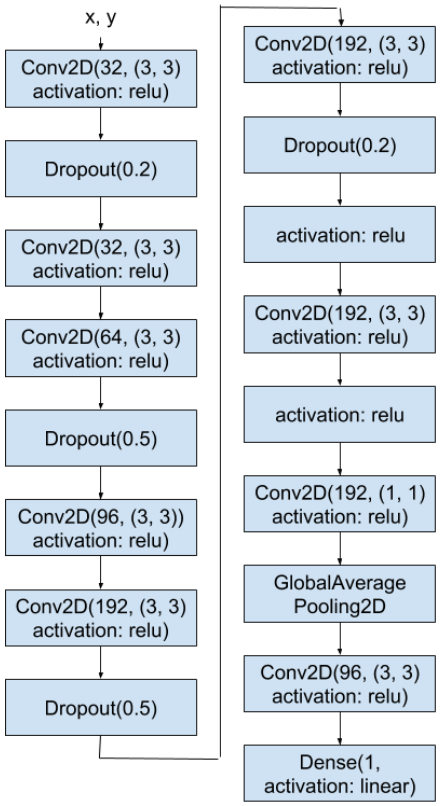

Figure 25. Architecture of the CNN regression network.

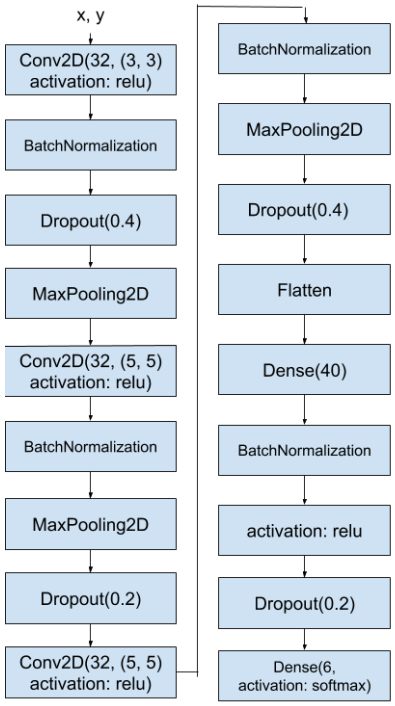

Figure 26. Architecture of the CNN classification network.

7.1.1 Analytic solution

The analytic solution solves the problem accurately, as expected. However, it would only work for this specific case. It cannot handle noisy data and would not work if the ball is not visible. Therefore this is not an approach that works well for the given problem, since the data will be noisy and the ball can be obscured by a player. Also the accuracy of the ball size could be hard to extract from the real data since it depends on the resolution of the images.

7.1.2 Fully connected networks using coordinates from a trajectory

As can be seen by comparing Figure 7 and 8 the classification using a fully connected network worked well. The points that have a greater uncertainty in getting correctly classified is the ones close to the boarder between classes. These points are expected to have a lower accuracy and does not affect the overall result significantly.

By making the network architecture larger the accuracy was slightly increased. But since the result was still very close to the previous network, the extra depth and complexity of the network did not seem to be motivated.

From Figure 11 we can establish that the network for regression worked well. The predictions are very close to the ground truth.

When adding noise to the ball size both the classifier and the regression performed worse. This is expected since the network probably solves the problem in a similar way as the analytic solution.

7.2. Solutions without using the ball size

The methods tested without the use of the ball size as input have varying results. One of the methods were able to provide a satisfactory result, proving that it is possible to provide a solution without the dependency of the ball size.

7.2.1 Fully connected networks using coordinates from a trajectory

When removing the ball size from the input both the classifier and the regression performed worse compared to when the ball size was included. This is expected since the network have less information to use and proves that the ball size is necessary for this type of network in the same way as it is necessary in the analytic solution.

7.2.2 Networks using time dependencies

The classification using networks with LSTM-layers worked well. There were some problems with trajectories with low heights, see for example the difference between Figure 17 and 18 in the lower left corner. The ball should be classified as ground according to the ground truth image but is classified as air. The network using GRU appear to have the same problem. It also has some problems with the long trajectory at the top of the image.

The prediction worked for the RNN's when the data used for training and validation was consisting of sequences of coordinates, and did not work when the data was just one point. This shows that the time dependency is critical for this solution. Another result which validates the time dependency as critical is the result for the fully connected networks when trained on data without ball size.

The prediction made with LSTM networks with noisy data works almost as well as with data without noise. Comparing Figure 18 and Figure 20, they perform better on different parts of the trajectories. The network trained on noisy data could predict some ground in the lower left corner where the network trained on data without noise failed.

The models seems to be robust to using smaller dataset sizes. This means that less data would have to be annotated if implemented using real data, saving time and reducing costs.

7.2.3 Networks with image input

As can be seen in Figure 21 and Figure 22, the network works well for for the first interval but not as well for the second. This might be due to that the network is well trained for some positions in the image and unfamiliar with other positions. Overall, the regression networks were able to get somewhat good result and have potential to get better if training with more data and fine tuning the parameters.

The networks that were tried for solving the problem by classification did not give any good results.

This is probably due to the small amount of training data and this can lead to overfitting. Several network architectures were tested but none of these gave a good result.

The biggest drawback for the networks using images as input is probably the size of the data set. We were able to train the network with a maximum of 3000 images but any larger input vector would make the memory limit to be reached. A basic rule of training networks is that there can never be too much data. One possible approach to be able to use a larger training data set is to eliminate one RGB-layer of the 4D images.

8. Conclusions

It is possible to determine the height of a ball without using its size, using a non-analytical method with a high accuracy. The best results were given when using the RNN's and that is probably because of the extra information given by the time dependence. A CNN regression is a possible way of getting the height directly from image input, and it is probably possible to get good results if a larger data set could be used for training.

For this specific problem, it is recommended to use coordinates rather than images since these give a more accurate estimation of the height of the ball and require less computing power.

9. Future work

There are many more adaptations and tests that can be done on the datasets than what has been represented here. Some of the suggestions are implementing CNN with LSTM in order to retrieve a network that can predict data without a delay in the output. It is also possible to use pre-trained networks and fine tune some parameters to make it fit wanted data. There are endless combinations of layer structures and parameters that can be tweaked, it is not guaranteed that the optimal networks have been obtained in this project.

References

[1] Google colaboratory . https://colab.research.google.com/notebooks/welcome.ipynb.

[2] Jupyter. https://jupyter.org.

[3] Unity. https://unity3d.com/.

[4] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen. *Recurrent Neural Networks for Short-Term Load Forecasting*. 2017. https://link.springer.com/content/pdf/10.1007%2F978-3-319-70338-1.pdf.

[5] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, and S. Tandon. *UFLDL Tutorial - Convolutional Neural Network*. http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/.