RoboCup
The Visual Referee Challenge

Daniël Vermaas

Layout: typeset by the author using LaTeX.
Cover illustration: Alexander Scheuber/Getty Images, processed with BlazePose.

RoboCup
The Visual Referee Challenge

Utilising BlazePose to interpret referee gestures

Daniël Vermaas
12208698

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

July 1st, 2022

Abstract

The RoboCup is a competition wherein teams all around the world develop autonomous robots that play soccer. While playing this game may be easy for humans, it poses countless challenges for these robots. One such challenge is the 'Visual Referee Challenge', which consists of understanding the referee. This thesis aims to resolve this challenge, creating a system which detects referee poses accurately. Solving this challenge would be helpful to improve upon the capabilities of robots in the RoboCup.

An additional level of difficulty comes in the form of the Standard Platform League (SPL). In this league all teams are required to use Nao robots. These robots do not poses advanced cameras or microphones, so a solution must work on hardware similar to these specifications. Our approach combines whistle detection using features such as Fast Fourier Transform (FFT), with pose estimation using BlazePose. Both of these sub-problems are subsequently trained on neural networks in order to classify whistle sounds and referee poses. This method allows for accurate results using low-end hardware.

Results showed that the whistle detection classifier attained great performance in all configurations, giving around 96% accuracy on average. Pose estimation gave more varied results, with the best model achieving 99.3% accuracy.

Acknowledgements

I would like to extend my sincere gratitude towards Dr. Arnoud Visser, who gave great insight and guidance during the creation of this thesis. These same thanks also extend to the Dutch Nao team, who clarified many technical aspects of their process and codebase. I'd like to acknowledge Dr. S. van Splunter for coordinating the course, and finally Shane Patrick for the formal English writing classes.

Contents

Chapter 1

Introduction

1.1 Context and Topic

In the year 2050, a team of humanoid autonomous robots will be fielded against the then current world champions in an official FIFA match (Kitano and Asada, 1998). At least that is the goal of the RoboCup. Since 1997, teams around the globe have competed against each other in various robot soccer leagues, all of which aim to further the research in robotics and artificial intelligence (Ferrein and Steinbauer, 2016). However, one of the problems that still needs to be solved is interpreting the referee, also known as 'the 'visual referee challenge' [1]. This challenge consists of accurately observing the referee and understanding their intent.

There are multiple avenues to approach this which have proven effective, one big example of this is the Kinect sensor (Kohli and Shotton, 2013). While methods like this would work with great accuracy, they would also go against the spirit of the competition. The end goal is to accomplish the objectives without external tracking objects, or complicated camera setups. Therefore, this research will focus on computationally inexpensive on-device pose estimation, using Blaze Pose (Bazarevsky et al., 2020).

Another way in which the referee will communicate with players is with their whistle. Detection of this is also instrumental for the robots, as it will signal when they need to pay attention to the referee, much like the wake-words of AI assistants. There has been great development pertaining to this in the RoboCup, Backer et al. (2014) and Fürtig et al. (2017) for example, which primarily achieve this by decomposing audio into audio-features.

Combining these two separate systems should lead to a system that will be able to accurately derive rulings from referee's when instructed to do so (by the whistle).

[1] https://spl.robocup.org/wp-content/uploads/SPL-Rules-2022.pdf#page=51

1.2 Scope

Some of the earliest work on whistle detection in the RoboCup domain can be found in Backer et al. (2014). They utilised many features, which all performed well in whistle detection. Utilising features such as Fast Fourier Transform (FFT) and Mel Frequency Cepstral Coefficients (MFCC's) gave high accuracy results of 98%, but no accuracy on live detection is mentioned. Pauli (2016) aimed to improve on this, by implementing denoising, and focusing on the live detection aspect.

Pose estimation in the RoboCup Standard Platform League (SPL) has less previous work, given that this part of the challenge is new this year. There are however quite some recent advancements in pose estimation which may prove useful. Primarily Bazarevsky et al. (2020), which provides accurate pose estimation using low end hardware.

1.3 Research Question

This thesis aims to provide [2] an accurate solution to the Visual Referee Challenge , whilst operating within the restrictions posed by the SPL. The accuracy of both the whistle detection and pose estimation should work on hardware of similar or lower specifications of Nao robots, ensuring that the research could be adapted in said RoboCup league.

1.4 Structure

The structure in which this research will answer the research question is as follows: Chapter 2 will provide the foundation of previous work. It will describe previous findings in said work and summarise these findings as to find a general idea of the problem. In Chapter 3 will focus on the methods used in creating the classifiers, such as the dataset creation, feature extraction and neural networks used. Chapter 4 will discuss the results of the research in the previous chapter, utilising a plethora of data visualisations and explanations. After which Chapter 5 will form a conclusion, discussing the results further. Ended by Chapter 6, which will discuss potential avenues for further development and improvement within the Visual Referee Challenge.

[2] `https://github.com/dvermaas/RefereeChallenge`

Chapter 2

Theory

This chapter will discuss the previous work done in the fields of both sound detection and pose estimation, within their respective paragraphs.

2.1 Audio Analysis

2.1.1 Audio Features

A great starting point for whistle detection is the field of Music Information Retrieval (MIR). It focuses on the extraction of audio features, as discussed in Futrelle and Downie (2002). This is also helpful when attempting to detect sounds like whistles. Backer et al. (2014) laid a lot of groundwork implementing whistle detection for usage in the RoboCup, like Fast Fourier Transform and Mel Frequency Cepstral Coefficients. On top of this numerous classifiers were used on these features. They attained impressive accuracies of around 98% within their dataset. The next two sections will explore these features in further detail.

2.1.2 Fast Fourier Transform

Fast Fourier Transform (FFT) is a faster way to calculate the discrete Fourier transform. Sound is usually composed of lots of individual sources, each source having their own frequency. FFT can be utilised to convert this sound back into a spectogram, which shows the frequencies that all the original sources produced. This method ideally picks up the frequency of the whistle, making FFT a valuable source of information for classifying various audio.

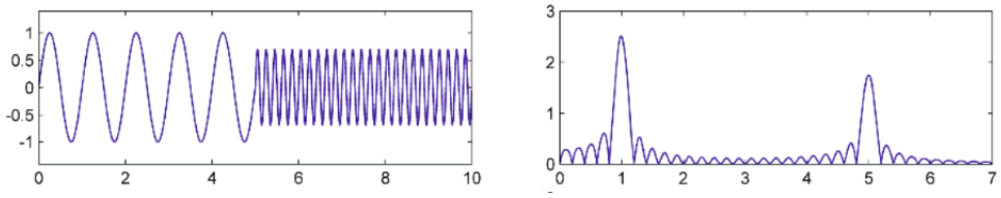


Figure 2.1: Visualisation of FFT

In Fig 2.1 one can see an audio signal that changes frequency at timestep 5. In the FFT at the right those two frequencies are distinguishable with two high peaks at frequency 1 and 5, although also many higher and lower tones can be seen. One can imagine that whistle detection would be elementary in this case, if the whistle produced one of the two peaks.

2.1.3 Mel Frequency Cepstral Coefficients

While the values that the FFT algorithm produces are understandable to the ways humans perceive sound, the same cannot be said about Mel Frequency Cepstral Coefficients (MFCC's). MFCC's may not have a clear definition, but they work well in many applications. MIR utilises MFCC to represent timbre for example, and MFCC is also used for speech recognition. These properties also translate in sound detection, as tested by Backer et al. (2014).

2.1.4 Current Solutions

There are various technical reports of note, which discuss their approaches to whistle detection. Hall et al. (2015) utilises FFT in their solution. They also implement a toggleable detector, in order to reduce their CPU usage. The Dutch Nao team technical report of Wiggers et al. (2020) only briefly discusses their current whistle detection implementation. But it does go over the details about how their implementation functions. It mentions the usage of FFT, after which analysis is done over the results. High peaks in certain frequencies would constitute whistle sounds. These solutions add to the various other works using FFT, suggesting that this method should be able to be a good foundation for accurate whistle detection.

2.1.5 Live sound detection

As previously stated Backer et al. (2014) achieved 98% accuracy in whistle detection. While high, it did not mention accuracy in live detection. Pauli (2016) aimed to build upon this previous work by implementing live detection of whistles.

The approach closely resembled that of Backer et al. (2014), but added denoising to the process. This is a sensible improvement, due to the fact that the RoboCup championship is a noisy event, with noise from spectators and other soccer fields. Another improvement Pauli (2016) makes is testing several different whistles. They do not state dataset validation accuracy, but do mention their live accuracy results. Their tests reportedly achieved between 60% and 90% accuracy, depending on the whistle used.

2.2 Pose estimation

2.2.1 Previous Approaches

While the challenge is new this year to the RoboCup SPL, pose estimation has been a well researched topic. Grest et al. (2009) compares several of these methods in 2D as well as 3D. Approaches like POSIT, which is a part of the open-cv library, could provide a useful framework in solving the challenge. Another approach is presented by Toshev and Szegedy (2014), they propose that deep neural networks could provide a more accurate solution to the pose estimation problem.

2.2.2 Separating Colors

The challenge explicitly mentions that the referee must wear red gloves, plus a white and black shirt. One way to play into these requirements is by utilising methods mentioned in Nunez et al. (2008). It describes algorithms that segment based on colors, which allows them to differentiate between both teams and the referee. Relying upon the color of shirts and gloves feels like a gimmick, which likely will be removed in further iterations of the challenge. However, it may prove useful if general pose-estimation cannot attain reasonable accuracy by itself.

2.2.3 BlazePose

BlazePose is a pose estimation model created by Google (Bazarevsky et al., 2020). It is a state of the art, lightweight estimator. This means that it can run on phones, which should make it a great fit for the limited processing power of Nao robots. BlazePose utilises a neural network to estimate poses, using heatmap, offset, and regression methods to achieve their great results. The model can be expanded to read the face or hands more in depth, meaning it could be expanded to detect more complicated referee gestures in the future. Adopting this framework also means that the need for red gloves should not be necessary. Making the model more resilient to potential dress code changes in the official SPL rules.

Chapter 3

Method

This chapter contains the methods used to build both classifiers. It will go over the dataset (creation), feature extraction, classifier models and the live detection element.

3.1 Audio Analysis

3.1.1 Dataset Details

The initial plan was to record a brand new dataset. But it turned out that the Nao SPL league already had an existing dataset [1] listed on their site. The dataset consists of ten recordings, having a run-time of one hour and 22 minutes. Some of the recordings are of full matches, while others are whistle detection tests. In total there are 51 whistles in the dataset, having a duration of one second on average. The dataset is very true to the likely competition environment of the visual referee challenge, given that it has been recorded on location. The dataset is recorded using the microphones of Nao robots and has a lot of background noise, which is expected at the RoboCup.

3.1.2 Dataset Balancing

While the dataset contains a lot of useful information, it also is very skewed towards non-whistle sounds. Balancing the dataset can be done in multiple ways. This study proposes two strategies to resolve this issue, making it a variable to be tested in the results. The data is represented by a list of frame indices that indicate where the whistles happen in each audio file (see Fig 3.1). This is converted to a list of

[1] Dataset: `https://sibylle.informatik.uni-bremen.de/public/datasets/whistle-2017/`.

frame indices where the change in classes happens. This is enough information, due to the fact that there are only two classes. At the end of the process all of the labelled audio-fragments are ready to be turned into features.

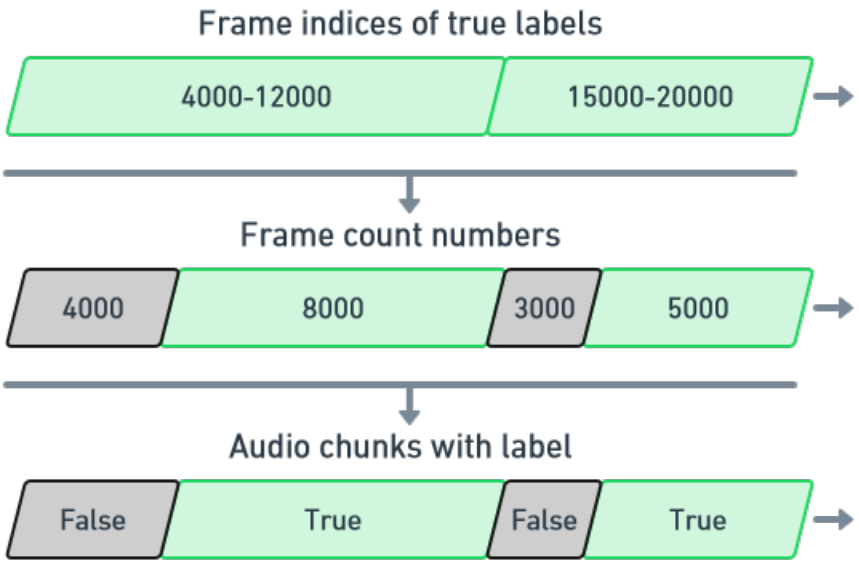


Figure 3.1: Visualisation of chunk gathering for the chunk processor

For each audio segment, a sample processor is executed. This processor converts audio into 50 millisecond samples. The processor also determines how staggered consecutive samples are. A value of 1/5 would mean samples would overlap like shown in figure 3.2, and a value of 2 would mean that a single sample length (50ms) would be left between consecutive samples. Both balancing approaches process all of the true segments with an overlap value of 1/5, but take different approaches to reduce false labels.

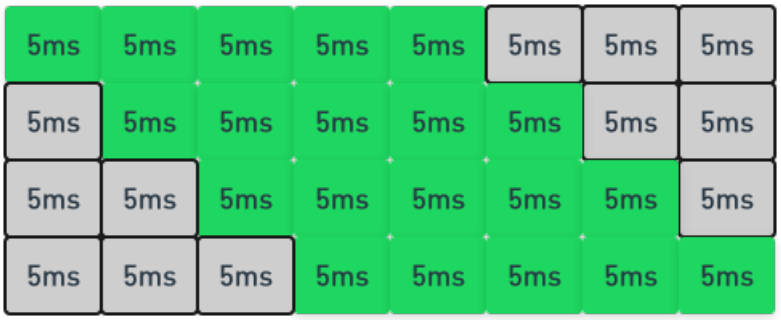


Figure 3.2: Visualisation of chunk processor staggering

The first balancing version iterates over all true segments. For each true segment, it calculates the number of frames within. It then grabs two segments of this length just before and after the true segment as the negative samples. It processes both of these additional segments using the exact same overlap value of the true labels, which is 1/5. This approach is a simple but effective way to achieve a 2:1 true false ratio. The general idea behind this balancing is that it puts emphasis on the borders between the negative and positive labels. So this method should in theory allow the classifier to learn what defines the transition between labels.

Version two takes a different approach to balance the data. It utilises two different window functions to achieve balance. This method uses the same 1/5 overlap for true values, while using 10 overlap for false values, meaning that after each sample a nine sample (450ms) gap is left. This discrepancy balances the two labels out, again resulting in a 2:1 label ratio. The philosophy behind this method is to spread out the negative samples more evenly across the dataset. This makes sure that the negative labels of the data are more diverse, theoretically resulting in a more robust classifier.

3.1.3 Feature extraction

The first Feature extraction layer takes care of denoising. Denoising is a variable that is being tested, and is therefore an optional step which takes place before future processing happens. Noise-reduction is achieved by using the python library called noisereduce (Sainburg et al., 2020).

After this step feature extraction is twofold. The first method is FFT. Each sample from the chunk processor is fed into the FFT algorithm, from the librosa

library (McFee et al., 2015). This produces a 2D array. Then the absolute values are taken from this array, after which the mean is calculated. This returns a 1D array that can be stored in a CSV file.

The second method used is MFCC. Each sample of the chunk processor gets fed into the MFCC algorithm, also from the librosa library. Then the mean is taken of this 2D array, reducing it to 1D and stored in CSV format.

3.1.4 Classifier

In all possible configurations the selected features are used as inputs in a neural network using Keras (version 2.3.1), a TensorFlow wrapper. This model consists of the input layer, one hidden layer containing 128 neurons, followed by a binary output layer with a single neuron, which gives a value between 1 and 0. A value of 0.5 or higher will classify as a whistle sound.

3.1.5 Live Detection

For live detection, pyaudio is used to read raw audio input. Input is read in 50ms buffers, which ensures that the live data is in the exact same format as the training data. The further setup is much like the process outlined in (Pauli, 2016). It repeats all of the aforementioned steps, resulting in a true or false output at a rate of 20Hz (see Fig 3.3).

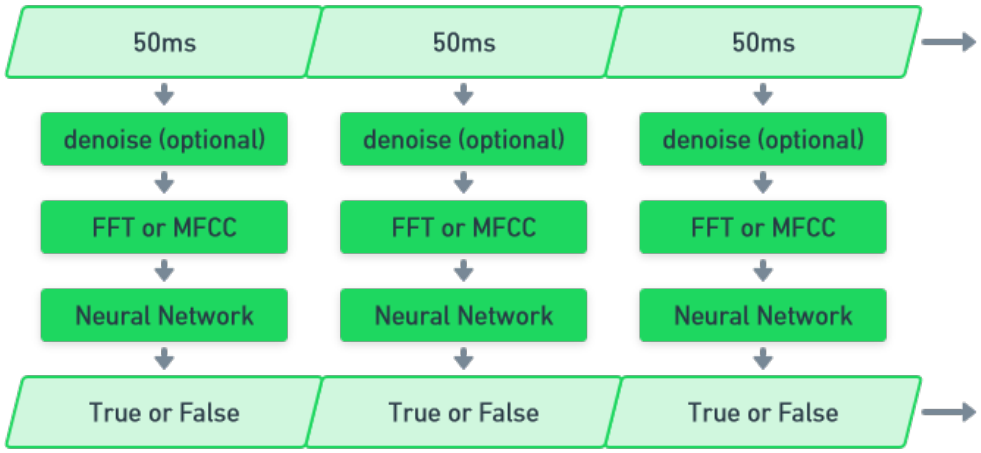


Figure 3.3: Visualisation of live detection process

Just using the output at every 50ms is a bit random at times, because some

outliers do occur. In order to combat this, the results are pushed into a queue. This makes sure that the last ten measurements are taken into account in the final prediction. When eight out of the ten samples are true a positive classification is made. Whistles take on 1000ms on average, so a 500ms buffer is within reason.

3.1.6 K-fold Validation

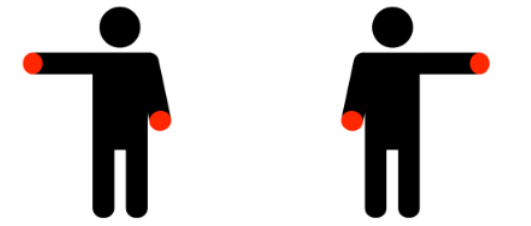
In order to evaluate the results, K-fold validation was used, five being chosen as the K-value. This means that validation is done by training on 80% of the data, and then validated for accuracy on the remaining 20%. These 20% validation partitions get cycled around, meaning that each partition will get validated, by the other 80%. This ensures that there is not a lucky or unlucky validation partition selected, making sure that the results are not influenced unfairly.

Another validation method used are confusion matrices. These matrices allow for more information on where the actual mistakes of the classifier are occurring. For live whistle detection validation a "FOX 40 CLASSIC TM" whistle was used. It is a whistle that is widely used in various sports, including the RoboCup. There are however no regulations pertaining to whistles, so whistle choice is up to the referee preference.

3.2 Pose Estimation

3.2.1 Dataset Details

Unlike whistle detection, pose estimation did not have any available datasets to use. This is due to the fact that the challenge is new this year, so no previous work has been published about the specific challenge. This meant that for this research a dataset had to be created from the ground up.



(a) Kick-in ⟨blue⟩ Team (b) Kick-in ⟨red⟩ Team

Figure 3.4: Example of team-specific pose

The challenge gives six poses which need to be recognised. Five of these poses are team specific, like figure 3.4 for example. This increases the pose count to eleven in total. In order to create the dataset, 64 photo's were recorded for each pose. Then these images were mirrored to create 64 additional pictures for each pose, making the total 128 samples per pose. The images feature slight distance and rotation variation in order to make the pose estimation more robust.

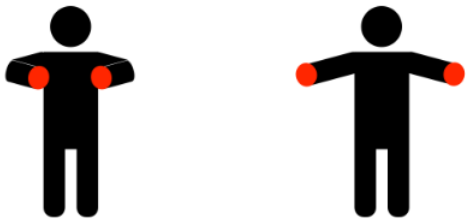


Figure 3.5: The 11th pose, which contains horizontal movement

The last pose, called full-time (see Fig 3.5), has movement associated with the pose. This is not something that the detection model can 'see', due to the fact it looks at a single frame at a time. In order to make this pose work like the others, samples were taken at various intervals of the animation. Therefore the classifier should be able to detect the pose at any point in the arm movement. Appendix A contains all poses for further reference.

3.2.2 Feature extraction

Feature extraction makes use of the BlazePose pose estimator. The algorithm takes in an image, and returns a skeleton, comprised of 33 connected points called landmarks as seen in Fig 3.6. These landmarks all have coordinates in 3D space. This pose estimation process is done on each picture of each pose, after which the coordinates and label are stored within a CSV file. This data by itself is enough to train classifiers on, which will be called the default model. On top of this multiple additional feature processors are tested, which will be discussed in the next section.

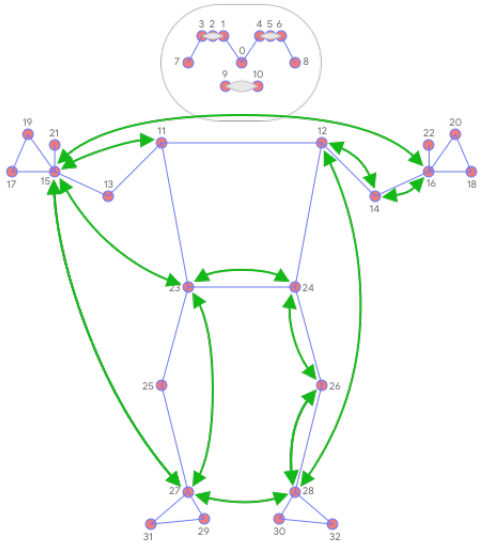


Figure 3.6: Visualisation of the landmarks BlazePose provides

3.2.3 Feature enrichment

In order to improve upon the base features, three ways were devised to improve accuracy. These methods were aimed at resolving multiple issues with the regular data. Both distance and different body-types could decrease accuracy. And additionally, only the arms matter for the estimating pose. Crossed legs or head movement should not be taken into account. The following methods aim to resolve these issues.

Anchoring & Normalising

The raw coordinates can be improved by anchoring the origin of the coordinate system to one of the landmarks, the left hip in this case. This ensures that the location relative to the camera won't matter, due to the fact that all coordinates will move with this origin. Normalising the distances of all points in the coordinate system would also improve accuracy, making the classifier able to cope with different body sizes. This is achieved by dividing all landmark distances from the hip, by the width of the hips.

Angles

The most straightforward way to improve the features is by looking at the angles of the arms. By calculating lines between the elbow and wrist as well as elbow

and shoulder, It is possible to calculate the angle between the two lines. This is done for the wrist elbow and shoulder for both arms, resulting in six values. This does reduce the amount of data significantly. But these values should be the only data needed to make predictions, while also eliminating the need for normalisation, since the angles would work for different arm lengths/distances.

Vectors

Improving upon the angles idea, the vector model aims to define angles in three directions. By calculating the lines as previously mentioned and then normalising the vectors, the model should have access to angles in three dimensions. Two lines have a lot of configurations in which the angles between them are the same, but the vector model does not suffer from this.

ML toolkit

The ML toolkit [2] is a Google solution which builds upon their BlazePose model (Bazarevsky et al., 2020). It encodes poses by calculating distances between landmarks, and normalising them. You can see these distance encodings visualised as green arrows in Fig 3.6. This model does not use a classifier like the others, instead opting for a k-nearest neighbours approach.

The ML toolkit also uses Exponential Moving Average (EMA) smoothing (Grebenkov and Serror, 2014). This smoothing takes a short history of frames into account and smooths out the data based on this. In EMA more recent results carry more weight as opposed to earlier samples. In practice the smoothing helps to make classifications flow more into each other, instead of predicting wildly different classes for each frame.

3.2.4 Classifier

In all possible configurations, the selected features are used as input in a neural network using Keras (version 2.3.1), a TensorFlow wrapper. This model consists of the input layer, one hidden layer containing 256 neurons, followed by a binary output layer with eleven neurons, one for each pose. The most activated neuron in this array will be the prediction of the model.

3.2.5 Live Detection

For live detection, open-cv (Bradski, 2000) is utilised to read raw camera input. On each frame, the pose estimator is executed. This in turn gives the same 33

[2]ML toolkit article: `https://google.github.io/mediapipe/solutions/pose.html`

landmark positions as the training data, which insures that the live data is in the exact same format as the training data. This data then gets fed into the selected post processor, after which they are used as classifier input. The classifier will then return the most likely label together with a certainty percentage (see Fig 3.7).

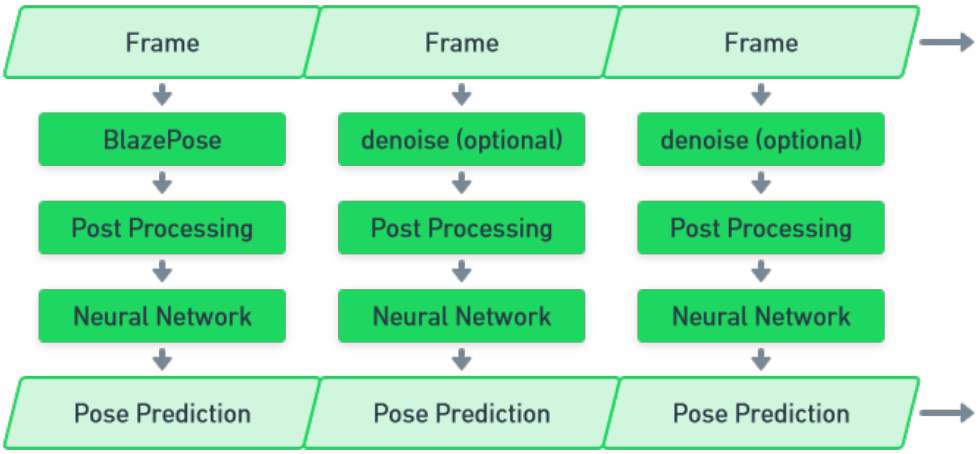


Figure 3.7: Visualisation of live detection process

Chapter 4

Results

4.1 Audio Analysis

4.1.1 K-fold Accuracy

Like mentioned in section 3.1.6, K-fold validation is utilised to test for accuracy on the whole dataset. The results have been split into two tables (see Table 4.1), one for each balancer version. On the x-axis are the two algorithm choices, and the y-axis is for the optional denoise layer (true means samples were denoised).

	FFT	MFCC
False	99.3%	**99.4%**
True	95.1%	97.8%

(a) Version 1 data

	FFT	MFCC
False	**99.2**%	99.1%
True	93.0%	95.8%

(b) Version 2 data

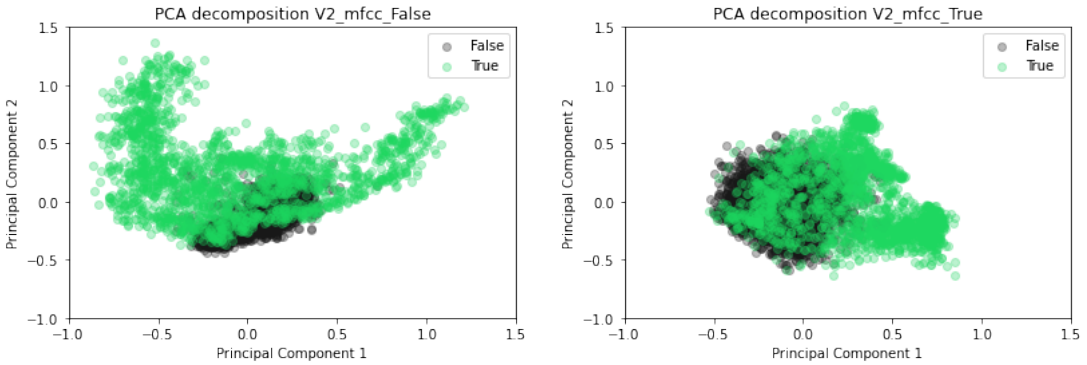
Table 4.1: K-fold accuracy for all configurations

When denoising is disabled, differences between the two versions are small. With denoising enabled differences of around 2% can be observed. The same also goes for comparing FFT with MFCC. They attain very similar results when denoising is disabled. When denoising is enabled MFCC does seem to slightly outperform FFT.

Denoising does have a general negative impact on validation accuracy. This implies that the denoising process causes some detail to be lost in the audio, an observation which will be explored further in the following section. Internal validation shows that all models achieve high results. While the difference between the two versions may therefore seem insignificant, the same does not have to hold true for live testing.

4.1.2 Dataset Decomposition

Principal Component Analysis (PCA) is a method that reduces the number of dimensions of the feature data. This enables the 2d plotting of said data to better understand how effective the features are at describing the labels. In figure4.1, both subfigures have a lot of overlap. This is not optimal, but not rare when performing PCA decomposition on data with a great number of dimensions.



(a) No denoising

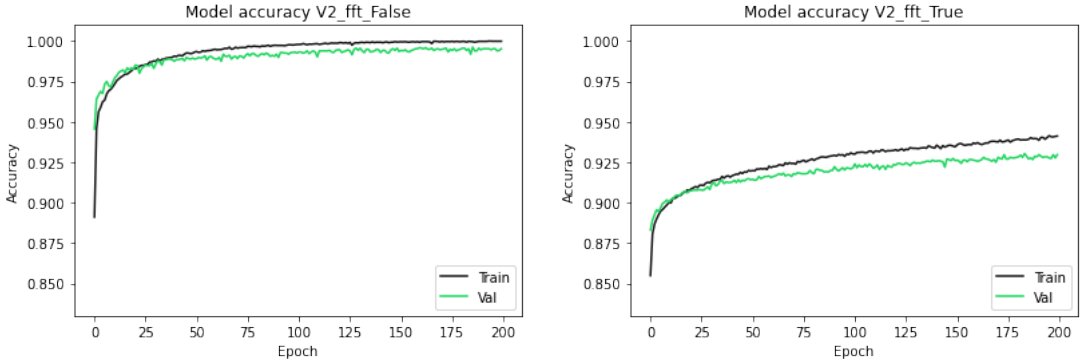
(b) Denoising

Figure 4.1: PCA decomposition difference when denoising is true/false

The signal without denoising (Fig 4.1a) displays a single dark cluster at the centre, as opposed to the green samples which are more spread out. The signal with denoising (Fig 4.1b) sees both clusters fully stacked on top of each other, making it significantly harder to find a boundary between the true and false labels.

4.1.3 Epoch Analysis

Each configuration is trained for 200 epochs, after which the best model weights are loaded. At each epoch the model is validated, to track how well the model is learning. Figure 4.2 shows the progression of two models, both using version 2 and FFT. Both models start with high accuracy, and show improvement over time.



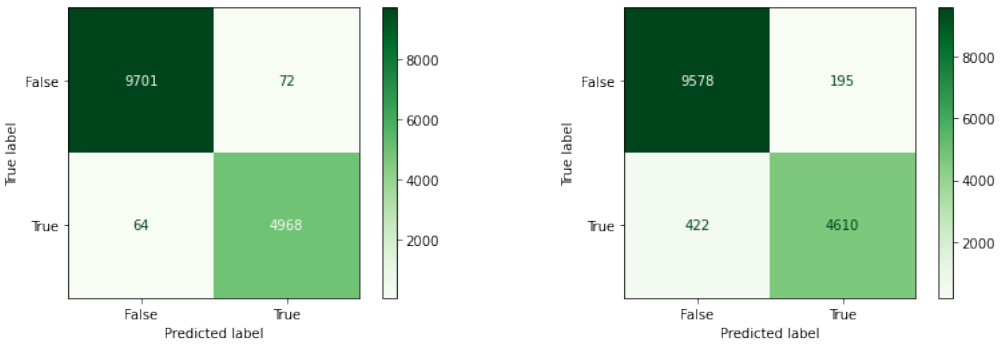
(a) No denoising (b) Denoising

Figure 4.2: Accuracy per epoch when denoising is true/false

Subfigure 4.2a Shows that train set accuracy quickly approaches 100%. The validation data does not fully grow in the same trajectory, but also achieves a high result. When looking at subfigure 4.2b, it is obvious that the trajectory will not reach 100%. Validation accuracy also lags behind more compared to Fig 4.2a. The differences between train- and validation accuracy are close in both subfigures, differing by less than 0.1%.

4.1.4 Confusion Matrices

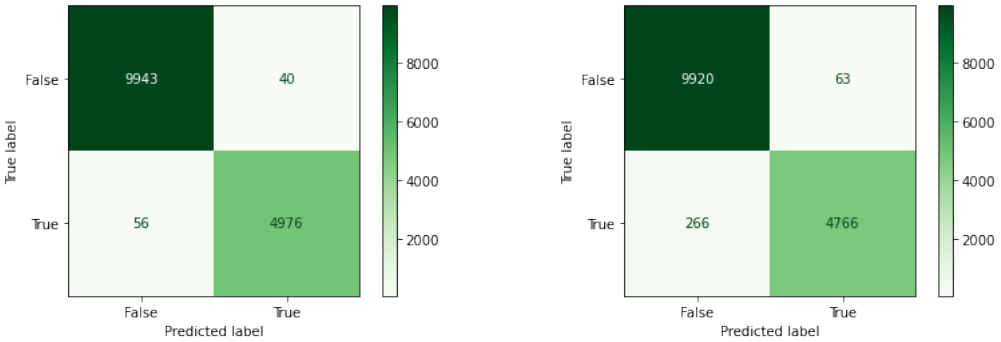
It is often the case that one label is harder to predict than others. Confusion matrices can help visualise these anomalies. Observing Fig 4.3, true positives- and negatives are both high and in a reasonable ratio. The false positives are interesting. They are quite close to the false negatives in Fig 4.3a, diverging a lot from the 2:1 ratio of false to true samples.



(a) V2, MFCC, no denoising (b) V2, MFCC, denoising

Figure 4.3: Confusion matrices

This same trend can be observed in Fig 4.4. Subfigure 4.4a also has a deviating ratio of false positives compared to false negatives. Fig 4.4b also distorts the 2:1 ratio, but in the opposite direction. One should keep in mind that the number of false positives and negatives are small in all matrices, making the deviations within reason.



(a) V1, MFCC, no denoising

(b) V1, MFCC, denoising

Figure 4.4: Confusion matrices

4.1.5 Live Validation

Live validation suffers from the fact that the environment cannot be fully controlled, unlike all of the previous validation. Real-time testing with the 'Fox 40 Whistle Classic' has shown that all models perform to acceptable standards. All models succeed in detecting all whistles, but with varying accuracy. For longer whistles it can even reach ten positive classifications in a row, but it will almost always attain 80% accuracy.

	No background noise	Background noise
Version 1 | FFT	**100**%	40.6%
Version 1 | MFCC	**100**%	90.6%
Version 2 | FFT	96.8%	**93.8**%
Version 2 | MFCC	84.4%	78.1%

Table 4.2: Live validation accuracy percentages (32 tests)

In table 4.2 results of the live detection test are displayed. The models were tested with and without background noise, and each model was tested 32 times. Each test consists of five seconds without whistle, then the whistle, followed by another five seconds. The model must detect the whistle with 80% certainty or

higher, while not getting any false positives before or after the whistle in order to score.

Without background noise, all models achieve great results, both version 1 models achieving a perfect score. The idea behind emphasising on the borders between the true and false labels seems to be sound when not dealing with background noise. With background noise this advantage disappears, instead version 2 performs the best when combined with FFT. The accuracy hit with background noise was mostly due to false positives. Version 1 FFT model was most susceptible to these mistakes, being very sensitive towards any noise.

4.2 Pose Estimation

4.2.1 K-fold Accuracy

Accuracy for pose estimation is measured in the same exact way as whistle detection, using K-fold validation. Table 4.3 shows that just using the landmarks, without further processing, gives very high results. The Anchor & Normalise process does increase accuracy slightly. Both the Angles and Vectors approaches fall far behind. They both can only achieve a third of the performance of the other two methods.

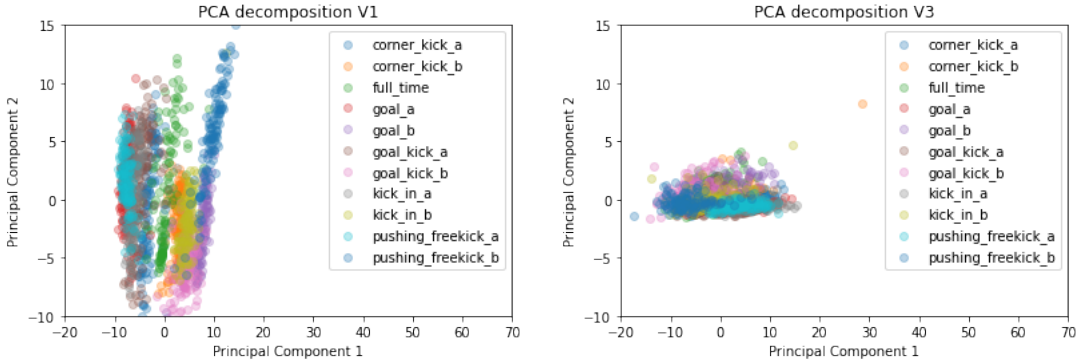
Default	Anchor	Angles	Vectors	ML Toolkit	ML Toolkit + EMA
98.9%	**99.3%**	33.2%	33.3%	96.6%*	59.5%*

Table 4.3: K-fold accuracy (*ML Toolkit measured without k-fold)

The ML Toolkit approaches give good results. The non EMA approach attains high accuracy. However, it is only validated on 20% test data once, due to limitations in the implementation. EMA smoothing performs much worse, but this is logical, since it smooths out movement between frames. This is useful in consecutive frames, but for validating isolated frames it harms results.

4.2.2 Dataset Decomposition

Figure 4.5 shows the principal component decomposition of the different models. In subfigure 4.5a, there are reasonable distances between the classes. This indicates that the data of the anchor model contains a lot of information for the classifier. A stark difference compared to subfigure 4.5b, where all labels have great overlap. These findings corroborate with the previous table, giving context to the poor accuracy results of the vector model.

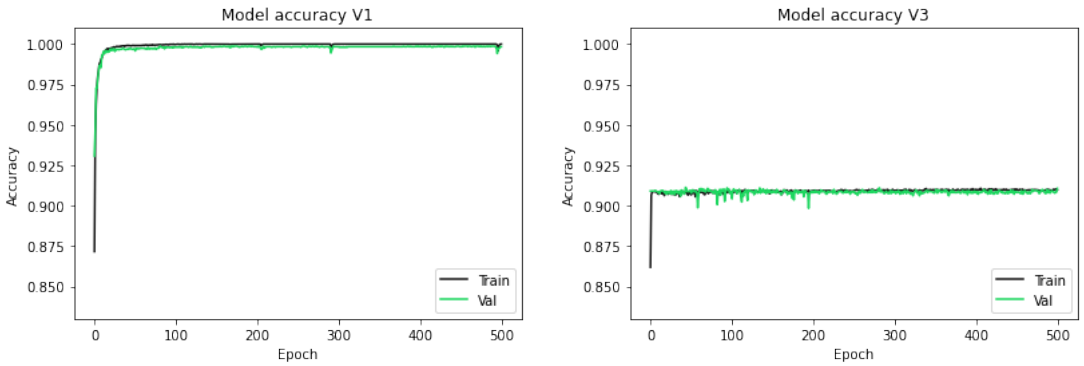


(a) Anchoring & Normalising model (b) Vector model

Figure 4.5: PCA decomposition

4.2.3 Epoch Analysis

Each model is trained for 500 epochs, after which the best model weights are loaded. Of note is that accuracy works very differently compared to the whistle detection method (Fig 4.6). Due to the fact that eleven poses are being evaluated, there are eleven outputs evaluated for accuracy. In order to get 100% accuracy for a given sample, the model must return ten negative neurons and a single positive one at the correct pose. This however also means that returning all negatives returns 91% accuracy.



(a) Anchoring & Normalising model (b) Vector model

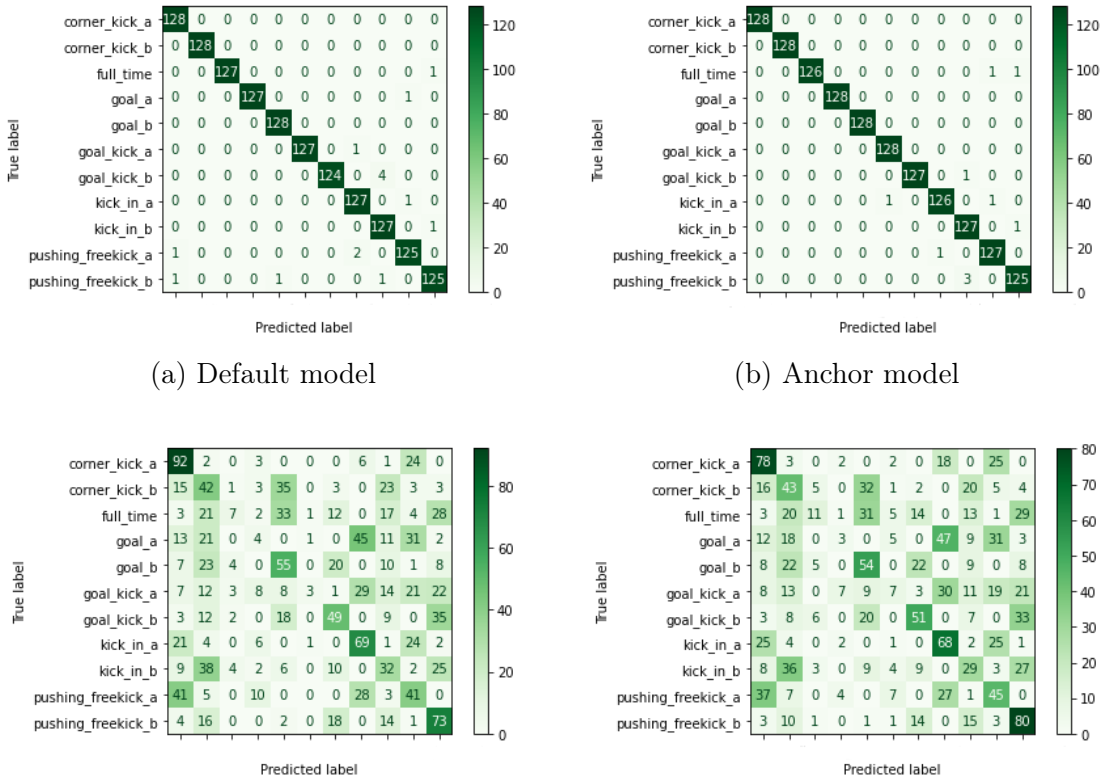
Figure 4.6: Accuracy per epoch

We can clearly see that this 91% is where the vector model strands in Fig 4.6b. No learning seems to be happening at all. Meanwhile in the anchor model learning is observable (see Fig 4.6a). Both train and validation accuracy rise quickly. The

default model plot is negligibly different from the anchor model and is therefore excluded. The same is true for the angle model, which is very similar to the vector model.

4.2.4 Confusion Matrices

In order to see which poses pose the largest challenge, confusion matrices prove useful. The y-axis in figure 4.7 are missing. The leftmost pose in both subfigures is the 'corner_kick_a', after which it follows the same order as the x-axis. In subfigures 4.7a & 4.7d all poses are predicted to near perfection. The 'pushing_freekick' poses seem to be the hardest, but only by small margins.



(a) Default model

(b) Anchor model

(c) Angle model

(d) Vector model

Figure 4.7: Confusion matrices of the first four models

Subfigure 4.7c & 4.7d are another story. It has much lower accuracy in most of the poses. It seems that the model does have a clear understanding of direction. When making errors, it often does pick the correct side (a or b). It also seems to favour one direction heavily for each pose. 'goal_a' for example has 55 correct

picks, while 'goal_b' only has 5. 'kick_in_a' is also twice more accurate then their mirrored counterpart.

4.2.5 Live Validation

Live validation suffers from the fact that the environment cannot be fully controlled, unlike all of the previous validation. Real-time testing using a webcam yielded poor results for both the angle and vector models. They would only predict 'kick_in_a'. The default and anchored models performed great. The default model worked well on the 'a' side, but performed worse on the other side. The anchor model improved on this a lot, only having difficulty with 'pushing_freekick_b' (see Fig A.5). The K-fold accuracy between the two models may be small, but in live detection the difference is very noticeable.

	Default	Anchor	Vector	ML Toolkit
kick in a	81.2%	100 %	100 %	100 %
kick in b	43.8%	100 %	0.0 %	100 %
goal kick a	96.9%	100 %	0.0 %	100 %
goal kick b	93.8%	93.8%	0.0 %	100 %
corner kick a	100 %	100 %	0.0 %	100 %
corner kick b	6.3 %	100 %	0.0 %	100 %
goal a	96.9%	100 %	0.0 %	100 %
goal b	0.0 %	100 %	0.0 %	100 %
pushing a	90.6%	96.9%	0.0 %	100 %
pushing b	53.1%	78.1%	0.0 %	100 %
full-time	81.2%	100 %	0.0 %	100 %

Table 4.4: Live validation accuracy percentages (32 tests)

In table 4.4 one can observe accuracies of multiple models. It is clear that the ML Toolkit solution outclasses all the competition, achieving a perfect score. The anchor model also performed well, achieving over 90% accuracy in all but one category.

Chapter 5

Conclusion

This thesis intended to combine whistle detection together with pose estimation. The first topic had lots of previous work associated with it. The results in this domain also show that all models closely match this previous work, achieving accuracies of up to 99%, like Backer et al. (2014). The fact that this accuracy is achievable while using relatively simple neural network architecture is surprising. The usage of a single hidden layer already gave excellent results. The robustness of the non-denoised detection models is also unexpected. Denoising clearly reduces the amount of useful information in the dataset, as illustrated by the PCA decomposition and K-fold accuracy.But most of the non-denoised models are robust enough to handle background noise.

Pose estimation gives unforeseen results as well. Filtering the data to only use the arms, which are the only real elements that matter for this challenge, proved to be a wrong approach. This data on its own simply was not enough to achieve adequate pose estimation. Both the angle and vector approach failed to exhibit any form of learning. The default model performed great, even though it theoretically had the most unpolished data. This is likely due to the fact that the dataset contains a lot of slight variance. Distance to the camera and angles between limbs were not the same on each picture, and therefore the default model could cope fairly well without having normalisation and adjusting for rotation.

The anchor & normalise model did not perform much better accuracy wise, but did significantly better in real time experiments. But in real-time testing none of the models came close to the ML Toolkit model. The distance based embedding proved to be a sound strategy. The main advantage it has is that it can utilise distances from seemingly irrelevant landmarks (like the ankles) to landmarks in both arms, giving multiple measurements for each important point in both arms.

Distances are also unaffected by rotations, providing even more advantages.

Results in both of the aforementioned fields have shown that 'The Visual Referee Challenge' is achievable using low-end hardware. The best models achieved over 95% accuracy in their respective task, and succeeded in translating this performance to live environments. Fulfilling all the requirements of the challenge.

Chapter 6

Discussion

While achieving great results, this research is limited primarily by data. Getting more full match audio recorded, and acquiring more pose samples would improve accuracy of both systems. Primarily for pose estimation 128 samples per pose is scarce. More variety in subjects would also improve the classifier, forcing it to adjust to different body types and proportions.

Another improvement could be achieved by utilising the full data of the audio features. Currently the 2D-arrays are converted into 1D-arrays by taking the means. Feeding the 2D-arrays directly into the neural networks could improve accuracy further. Adding more layers did not seem necessary using 1D-arrays, but this new data could lend itself to more complicated multi-layered neural networks.

Implementation of this model, or improvements thereof, into the Nao robots would be splendid. While limitations of these bots have been taken into account, it cannot be said for sure if they would perform as well when implemented fully into them. It would show how strained the CPU gets when running these live classifiers, and how well the software works when implemented into existing codebases.

Bibliography

Backer, N. W., Intelligentie, B. O. K., and Visser, A. (2014). Horn and whistle recognition techniques for nao robots. *Bachelor thesis, Universiteit van Amsterdam.*

Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., and Grundmann, M. (2020). Blazepose: On-device real-time body pose tracking. *CVPR Workshop on Computer Vision for Augmented and Virtual Reality, Seattle, WA, USA, 2020.*

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools.*

Ferrein, A. and Steinbauer, G. (2016). 20 years of robocup. *KI-Künstliche Intelligenz*, 30(3):225–232.

Fürtig, D.-I. A., Brast, J. C., Ditzel, S., Hammer, H.-J., Hess, T., Rinfreschi, K., Siegl, J.-M., Steiner, S., Weiglhofer, F., and Wörner, P. (2017). Team description for robocup 2017.

Futrelle, J. and Downie, J. S. (2002). Interdisciplinary communities and research issues in music information retrieval. In *ISMIR*, volume 2, pages 215–221.

Grebenkov, D. S. and Serror, J. (2014). Following a trend with an exponential moving average: Analytical results for a gaussian model. *Physica A: Statistical Mechanics and its Applications*, 394:288–303.

Grest, D., Petersen, T., and Krüger, V. (2009). A comparison of iterative 2d-3d pose estimation methods for real-time applications. In *Scandinavian Conference on Image Analysis*, pages 706–715. Springer.

Hall, B., Harris, S., Hengst, B., Liu, R., Ng, K., Pagnucco, M., Pearson, L., Sammut, C., and Schmidt, P. (2015). Robocup spl 2015 champion team paper. In *Robot Soccer World Cup*, pages 72–82. Springer.

Kitano, H. and Asada, M. (1998). The robocup humanoid challenge as the millennium challenge for advanced robotics. *Advanced Robotics*, 13(8):723–736.

Kohli, P. and Shotton, J. (2013). Key developments in human pose estimation for kinect. In *consumer depth cameras for computer vision*, pages 63–70. Springer.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O. (2015). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25.

Nunez, J. R., Facon, J., and de Souza Brito, A. (2008). Soccer video segmentation: referee and player detection. In *2008 15th International Conference on Systems, Signals and Image Processing*, pages 279–282. IEEE.

Pauli, T. (2016). Trillerpfeifenerkennung für einen humanoiden fußballroboter. Bachelor thesis, Freien Universität Berlin.

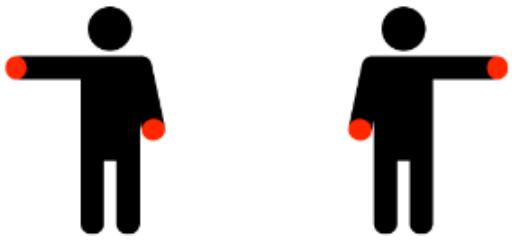
Sainburg, T., Thielk, M., and Gentner, T. Q. (2020). Finding, visualizing, and quantifying latent structure across diverse animal vocal repertoires. *PLoS computational biology*, 16(10):e1008228.

Toshev, A. and Szegedy, C. (2014). Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660.

Wiggers, T., van der Wal, D., Lekanne gezegd Deprez, H., Zwerink, W., and Kronemeijer, P. (2020). Dutch nao team. Technical report, Universiteit van Amsterdam.

Appendix A

All Pose Estimation Poses



(a) Kick-in ⟨blue⟩ Team (b) Kick-in ⟨red⟩ Team

Kick-in ⟨color⟩ Team. One-handed signal. One arm, extended horizontally in the direction of the half of the field corresponding to the team that receives the Kick-in Free Kick. That is, right arm extended for the "Blue team", and left arm extended for the "Red team". The non-signal hand is flat and motionless by the side of the body.

Figure A.1: Kick-in pose

(a) Goal Kick ⟨blue⟩ Team (b) Goal Kick ⟨red⟩ Team

Goal Kick ⟨color⟩ Team. One-handed signal. One arm, extended 45-degree *up* in the direction of the end of the field where the goal kick will occur. That is, right arm extended for the "Blue team", and left arm extended for the "Red team". The non-signal hand is flat and motionless by the side of the body.

Figure A.2: Goal Kick pose

(a) Corner Kick ⟨blue⟩ Team (on the half of the red team) (b) Corner Kick ⟨red⟩ Team (on the half of the blue team)

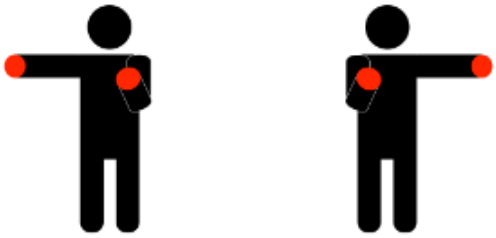
Corner Kick ⟨color⟩ Team. One-handed signal. One arm, extended 45-degree *down* in the direction of the team executing the corner kick. That is, right arm extended for the "Blue team" executing the corner kick on the "Red team's" side, and left arm extended for the "Red team" executing the corner kick on the "Blue team's" side. The non-signal hand is flat and motionless by the side of the body.

Figure A.3: Corner Kick pose

(a) Goal ⟨blue⟩ Team (b) Goal ⟨red⟩ Team

Goal ⟨color⟩ Team. Two-handed signal. One arm, extended pointing at the center circle. Other arm, extended horizontally in the direction of the half of the field corresponding to the team that scored the goal. That is, right arm extended for the "Blue team", and left arm extended for the "Red team".

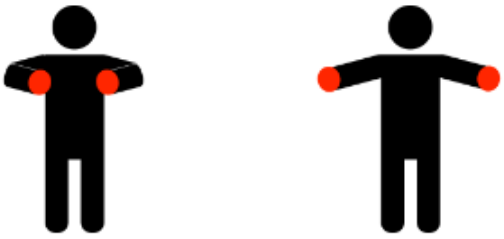
Figure A.4: Goal pose



(a) Pushing Free-kick ⟨blue⟩ Team because a red robot has pushed. (b) Pushing Free-kick ⟨red⟩ Team because a blue robot has pushed.

Pushing Free-kick ⟨color⟩ Team. Two-handed signal. One arm, vertical with bent elbow and palm facing in the direction of the extended arm. Other arm, extended horizontally in the direction of the half of the field corresponding to the team that is *executing* the Free-kick. That is, left arm extended for the "Red team", and right arm extended for the "Blue team".

Figure A.5: Pushing Free-kick pose



Full-Time. Dynamic two-handed signal. Both arms slowly move symmetrically inward and outwards on a horizontal plane, bending at the elbows. Note, for the purpose of this challenge, the whistle associated with this signal should be a *single* blow, unlike in normal SPL games.

Figure A.6: Full-Time pose