Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik
Lehrstuhl für Adaptive Systeme
Prof. Dr. Verena V. Hafner

Studienarbeit

# Analyse von RoboCup Spielen

## Erkennen und Lokalisieren von Nao Robotern

Autor: Stella Alice Schlotter
schlottb@informatik.hu-berlin.de

Betreuer: Prof. Dr. Verena V. Hafner

Berlin, March 13th, 2020

# Zusammenfassung

Die Durchführung von RoboCup Fußballspielen erfordert einen hohen Zeit- und Geldaufwand und hat trotzdem einen geringen wissenschaftlichen Nutzen. So lassen sich fundierte Aussagen über die Performance der Algorithmen nur schwer treffen. Besonders die Analyse komplexer Prozesse, wie Entscheidungsfindung eines Roboters oder das Teamverhalten gestaltet sich schwierig. Das Sammeln von mehr Daten bei den RoboCup Spielen soll helfen die Algorithmen besser zu analysieren, Fehler festzustellen und Verbesserungsmöglichkeiten zu finden. Ein System welches die Aufnahme und Analyse von RoboCup Daten unterstützt wurde bereits in mehreren Publikationen von dem Team Berlin United vorgestellt. In dieser Arbeit geht es um das Erkennen und lokalisieren der Roboter auf dem Feld basierend auf Videoaufnahmen einer Kamera am Spielfeldrand.

# Inhaltsverzeichnis

**1 Einleitung** **1**

**2 NaoTH Groundtruth System Überblick** **3**

**3 Objekterkennung mit Yolo** **6**

3.1 Der Yolov2 Algorithmus . . . . . 7

3.1.1 Anchor box clustering . . . . . 8

3.1.2 Loss Funktion . . . . . 9

3.1.3 Model Architektur . . . . . 9

3.2 Auswertung der Ballerkennung mit Yolo . . . . . 10

3.2.1 Laborkamera 1 . . . . . 10

3.2.2 Laborkamera 2 . . . . . 10

3.2.3 RoboCup 2018 . . . . . 11

3.3 Nao Roboter Erkennung . . . . . 11

3.3.1 Labor Kamera 1 . . . . . 11

3.3.2 Labor Kamera 2 . . . . . 11

3.3.3 RoboCup 2018 . . . . . 11

**4 Roboter Lokalisierung** **14**

4.1 Feldlinienkennung . . . . . 14

4.2 Bestimmung der Position der Kamera . . . . . 15

4.3 Lokalisierung der Roboter . . . . . 17

**5 Fazit** **20**

**Literaturverzeichnis** **22**

# 1 Einleitung

Die RoboCup Initiative wurde gegründet um Robotikforschung zu vergleichen. Zu diesem Zweck werden jährliche Veranstaltungen wie die RoboCup Weltmeisterschaft und zahlreiche lokale Veranstaltungen organisiert. Diese Veranstaltungen sind mit hohem finanziellen und zeitlichen Aufwand verbunden. Eine systematische Auswertung der einzelnen Spiele sowie eine Analyse über Meisterschaften hinweg findet nicht statt. Die Spiele dienen bisher den Teams vor allem als subjektive Einschätzung ihrer Arbeit. In vorangegangen Arbeiten hat sich das Team Berlin United - NaoTH mit der Erstellung und Deployments eines Low-Cost Groundtruth Systems beschäftigt. Dieses System wird ausführlich in [15] und [9] vorgestellt. Teile des Systems wurden bereits früher in [8, 13, 14] vorgestellt. In dieser Arbeit soll die Erkennung und Lokalisierung der Roboter in der Videoaufnahme des Spiels genauer analysiert werden.

Die Motivation hinter dieser Arbeit kommt aus der Analyse von Teamverhalten in Sportarten wie Basketball, American Football und Fußball. In diesen Sportarten ist es üblich das Spielgeschehen anhand der gemessenen Bewegungsdaten zu analysieren.

So nutzen beispielsweise [16] die Bewegungsdaten von Basketballspielern um Ereignisse wie Passen und Würfe vorherzusagen. In [5] werden aus vergangenen Positionsdaten der Spieler die zukünftigen Position vorhergesagt mittels eines *Conditional Variational Autoencoders*.

Eine besonders interessanter Arbeit ist das Data-Driven Ghosting [7] von Disney Research. Die Autoren dieses Papers nutzen die Positionsdaten der Spieler zweier menschlichen Fußballmanschaften aus der englischen *Premiere League* um Verhaltensmodelle der Teams zu lernen. Es werden dabei die Daten einer ganzen Saison verwendet. Das gelernte Modell kann genutzt werden um was-wäre-wenn Fragen zu beantworten. Zum Beispiel hätte Mannschaft 1 ein Tor geschossen wenn ein Mitspieler woanders gestanden hätte. Um diese Forschung aus den Sportwissenschaften nutzbar für die RoboCup SPL Liga zu machen braucht es ein zuverlässiges Aufnahmesystem welches Roboterpositionsdaten aufnimmt.

Die vorliegende Arbeit ist wie folgt gegliedert: In Kapitel 2 wird der aktuelle Stand des *NaoTH Groundtruth Systems* vorgestellt. In Kapitel 3 wird die visuelle Erkennung der

Roboter in den Videoaufnahmen vorgestellt und evaluiert. In Kapitel 4 wird die Kameralokalisierung und anschließende Roboterlokalisierung vorgestellt und evaluiert.

# 2 NaoTH Groundtruth System Überblick

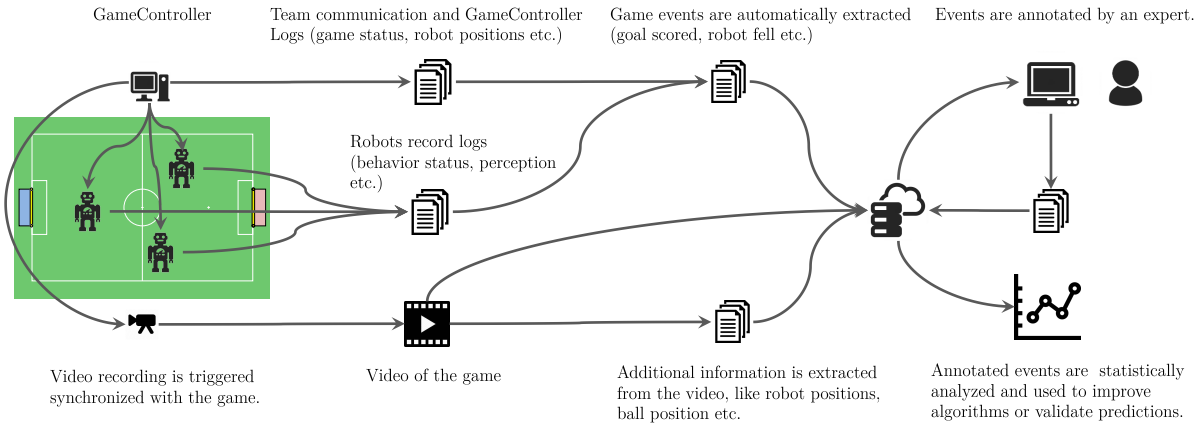



Abbildung 2.1: Darstellung des Datenflusses im NaoTH Groundtruth System

Zurzeit unterscheiden wir im System drei verschiedene Arten von Daten, die während eines Spiel aufgenommen werden: Das *Video* des Spiels, welches von einer Kamera am Spielfeldrand aufgenommen wird. Die *Netzwerk Kommunikation* der Roboter untereinander und mit dem *GameController PC*. Und abschließend die *Log Daten* die jeder Roboter generiert.

Die Aufnahme des Videos passiert zeitgleich mit dem GameController Signal für den Spielbeginn. Auf diese Art sind die TeamComm Logs und das Video automatisch synchronisiert. Im Team *Berlin United - NaoTH* sind die Roboter so programmiert, dass die Logs der individuellen Roboter auch erst ab Spielbeginn aufgezeichnet werden. Somit sind die individuellen Logs ebenfalls synchronisiert mit dem Video und den TeamComm Logs. Diese Automatische Aufnahme ist in [15] ausführlicher beschrieben.

In [9] wird ein Proof of Concept für das Lokalisieren und Verfolgen von Robotern mit einer monokularen Kamera vorgestellt. Die Funktionsweise soll in folgenden Kapiteln näher beschrieben werden.

Die Sicht der Kamera vom Spielfeldrand ist beispielhaft in Abbildung 2.2 zu sehen. Im Labor des Team Berlin United - NaoTH sind zwei Kameras permanent installiert. Diese

Abbildung 2.2: Sicht des Kamerasystems auf das Spielfeld beim RoboCup 2018

Kameras haben jeweils eine deutliche andere Sicht auf das Spielfeld. In Abbildung 2.3 und Abbildung 2.4 sind Beispielbilder der beiden Laborkameras zu sehen.

Abbildung 2.3: Beispiel Bild aus dem Blickwinkel der Laborkamera 1

Abbildung 2.4: Beispiel Bild aus dem Blickwinkel der Laborkamera 2

# 3 Objekterkennung mit Yolo

Der Yolo Algorithmus ist ein Objekterkennungsalgorithmus welcher besonders auf Geschwindigkeit optimiert wurde. Nach dem originalen Yolo Paper [10] wurden zwei weitere Varianten vorgestellt. Einmal *Yolov2* [11] und *Yolov3* [12]. Nach Aussagen der Autoren schlägt *Yolov2* alle anderen Neuronale Netze für Objekterkennung in der Inferenzzeit auf dem *PASCAL VOC 2007* Datensatz. Aus diesem Grund wurde *Yolov2* für diese Arbeit ausgewählt. Die Autoren des Paper haben folgende Vergleichstabelle für den *PASCAL VOC 2007* Datensatz in ihrem Paper vorgestellt. Alle Yolov2 Modelle sind die exakt selben mit den selben Gewichten. Nur die Inferenz geschieht auf unterschiedlich skalierten Bildern. Die Zeiten beziehen sich auf die Geforce GTX Titan X.

| Detection Frameworks | Train | mAP | FPS |
|---|---|---|---|
| Fast R-CNN | 2007+2012 | 70.0 | 0.5 |
| Faster R-CNN VGG-16 | 2007+2012 | 73.2 | 7 |
| Faster R-CNN ResNet | 2007+2012 | 76.4 | 5 |
| YOLO | 2007+2012 | 63.4 | 45 |
| SSD300 | 2007+2012 | 74.3 | 46 |
| SSD500 | 2007+2012 | 76.8 | 19 |
| YOLOv2 288x288 | 2007+2012 | 69.0 | 91 |
| YOLOv2 352x352 | 2007+2012 | 73.7 | 81 |
| YOLOv2 416x416 | 2007+2012 | 76.8 | 67 |
| YOLOv2 480x480 | 2007+2012 | 77.8 | 59 |
| YOLOv2 544x544 | 2007+2012 | **78.6** | 40 |

Tabelle 3.1: Yolov2 im Vergleich. Tabelle ist dem Yolov2 Paper [11] entnommen.

Im folgenden wird die Funktionsweise von Yolov2 vorgestellt. Da die Yolo Paper sehr lückenhaft sind wurden für die Erklärung der Funktionsweise sekundäre Quellen herangezogen. Vor allem verschiedene existierende Implementierungen [1], [2], [3] and [4].

## 3.1 Der Yolov2 Algorithmus

Der Yolo Algorithmus unterteilt das Bild in ein SxS Gitter. Für die Experimente in 3.2 und 3.3 wurde ein 13 mal 13 Gitter verwendet. Eine Beispielgittergröße ist in Abbildung 3.1 illustriert. Während des Trainings wird das Objekt der Zelle zugewiesen in welcher sich der Mittelpunkt der *Bounding Box* befindet. Für jede Zelle werden $B$ Bounding Boxen und $C$ Klassen vorhergesagt. Die Vorhersage der Bounding Box ist ein Vektor der Länge 5 mit den folgenden Einträgen: $p_b = (x, y, w, h, confidence)^T$ wobei x, y den Mittelpunkt der Bounding Box relativ zur Zelle repräsentiert. x und y werden auf 0 und 1 normalisiert. w,h ist die Größe der Bounding Box relativ zur Bildgröße. Ein Beispiel ist in 3.2 zu sehen.

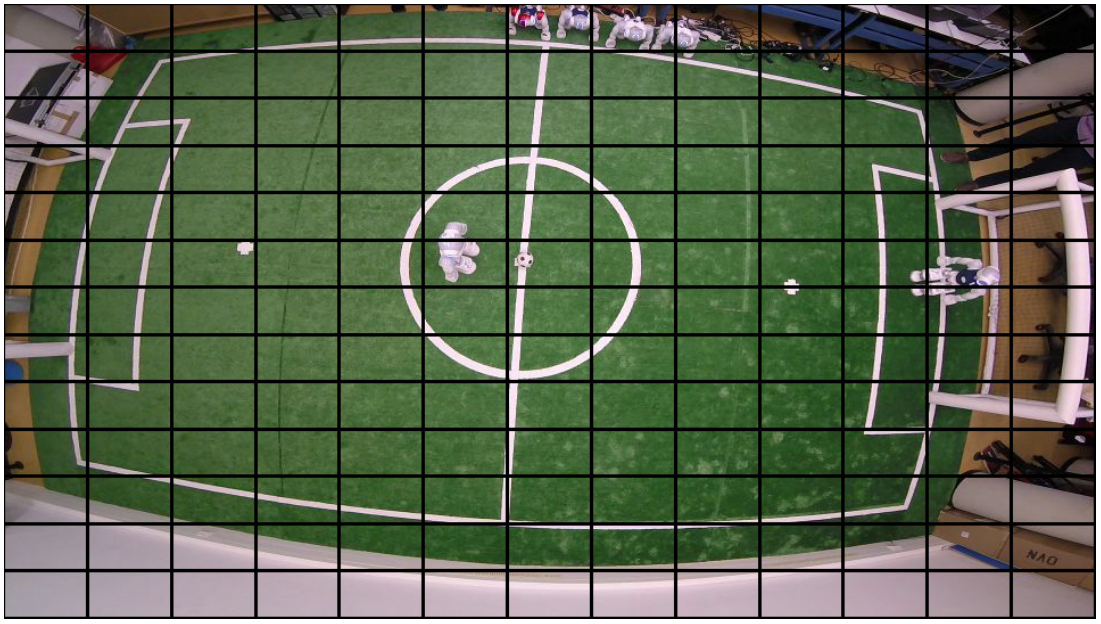
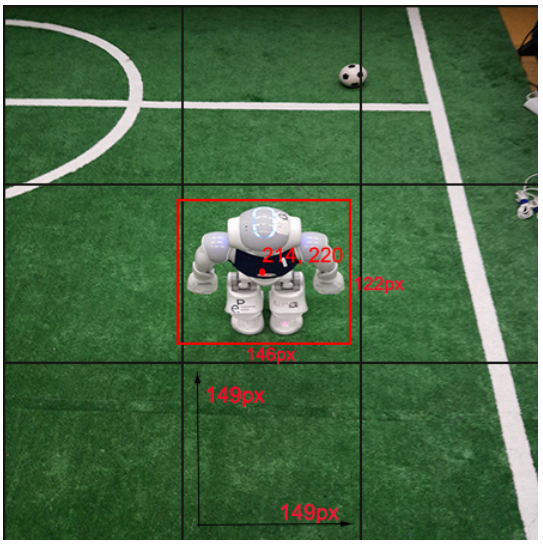



Abbildung 3.1: In dieser Arbeit wurde ein 13x13 Gitter benutzt. Dies entspricht dem Yolo Algorithmus wie er in [10] und [11] für die Erkennung auf dem PASCAL VOC Datenset vorgestellt wurde.

Der vorhergesagt Mittelpunkt ist (214, 220). Die Breite der Bounding Box ist 146px und die Höhe ist 122px. In diesem Beispiel hat eine Gridzelle eine Höhe und Breite von 149px.

Mit diesen Werten ergeben sich für die 4 Formparameter der Bounding Box folgende Werte:




$$x = \frac{214-149}{149} = 0.43$$
$$y = \frac{220-149}{149} = 0.47$$
$$w = \frac{146}{149} = 0.97$$
$$h = \frac{122}{149} = 0.81$$

Abbildung 3.2: Beispiel Yolo Bounding Box Parametrisierung

### 3.1.1 Anchor box clustering

Im Yolov2 Paper [11] werden sogenannte *Anchor Boxes* eingeführt. Diese sind mehrere vorher festgelegte Rechtecke. Im Falle von Yolov2 sind es standardmäßig 5 Rechtecke. Jedes dieser Rechtecke dient als *prior* für die Objekterkennung in einer Zelle. Es wird also für jeden Prior eine Erkennung pro Zelle durchgeführt. Die Autoren des Yolo Papers empfehlen die priors mit Hilfe des kmeans Algorithmus zu bestimmen. Dabei werden die Höhe und Breite aller Bounding Boxen im Trainigsdatenset zu n Clustern aufgeteilt. Die Mittelpunkte der Cluster stellen dann die priors dar. Das Clustering für die Bounding Boxen in meinem Trainingsset ist beispielhaft in 3.3 dargestellt. Während des Trainings wird für jede Zelle in der ein Mittelpunkt einer Bounding Box liegt ein Regressor auf einer Anchor Box gelernt welcher das Trainingsbeispiel lernen soll. Der Regressor wird dabei auf der Anchor Box gelernt welche die höchste IOU (Intersection over Union) mit der Bounding Box aus dem Training hat. Die IOU Metrik ist dabei wie folgt definiert:

$$IOU = \frac{|AnchorBox \cap BoundingBox|}{|AnchorBox \cup BoundingBox|} \tag{3.1}$$

Diese Metrik wird in der Literatur auch als Jaccard Index bezeichnet. Yolo lernt neben der Bounding Box Position und Dimension auch ein Konfidenz Wert für die letztendliche gefundene Bounding Box. Während des Trainings ist der Konfidenz Wert definiert als:

$$confidence = \begin{cases} IOU & \text{wenn Bounding Box in der Zelle ist} \\ 0 & sonst \end{cases} \tag{3.2}$$

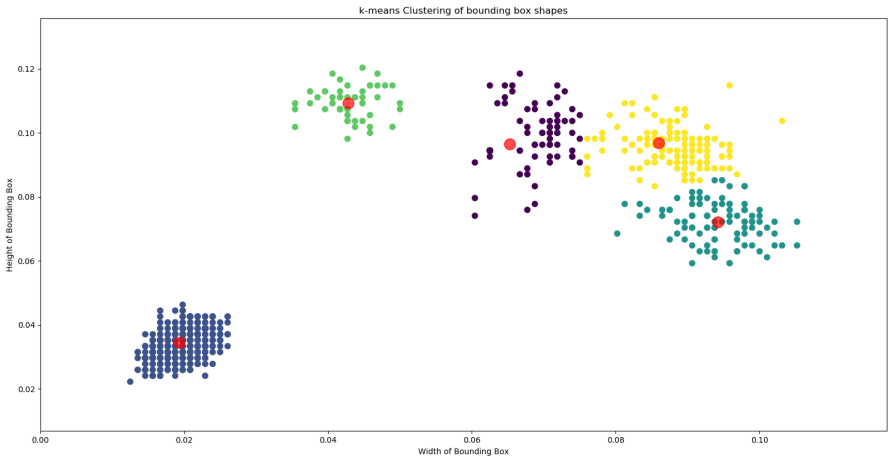



Abbildung 3.3: Clustering der Boundings Box Shapes im Trainingsset

### 3.1.2 Loss Funktion

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \tag{3.3}$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \tag{3.4}$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \tag{3.5}$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{3.6}$$

$$\tag{3.7}$$

In dieser Loss Funktion beschreiben $x_i$ und $y_i$ den Mittelpunkt einer Anchorbox und $h_i$ und $w_i$ sind die Höhe und Breite einer Anchorbox. $C_i$ ist der *confidence score* der detektierten Klasse. $p_i(c)$ ist der Klassifikationsloss. $\mathbb{1}_{ij}^{noobj}$ ist 1 wenn kein Objekt in der Zelle ist ansonsten ist $\mathbb{1}_{ij}^{noobj}$ gleich 0. $\mathbb{1}_{ij}^{obj}$ ist gleich 1 wenn eine Klasse in dieser Zelle erkannt wird, ansonsten 0. $w_i$ und $h_i$ stehen unter der Wurzel um den Fehler bei kleinen Bounding Boxen stärker zu bestrafen.

### 3.1.3 Model Architektur

In Abschnitt 3.1.3 ist das verwende Model Yolov2 dargestellt. Dieses Model entspricht dem Model welches als Yolov2 Netz in dem Repository der Yolo Autoren [1] zu finden ist. Im

[1] `https://github.com/pjreddie/darknet`

Paper ist nicht eindeutig geklärt ob die Metriken im Paper mit genau dieser Architektur entstanden sind. Die Autoren haben noch weitere ähnliche Modelle veröffentlicht. Für das in 3.2 beschriebene Netz wurden vortrainierte Gewichte [2] für die *Convolution* Schichten genutzt.

## 3.2 Auswertung der Ballerkennung mit Yolo

Wie in Abbildung 2.3 und Abbildung 2.4 abgebildet gibt es zwei stark unterschiedliche Sichtweisen auf das Laborfeld. Die Sicht der Kamera bei offizielen RoboCup Spielen ist ähnlich der zweiten Laborsicht. Es wurde für die ersten Experimente für jede Sicht die Ballerkennung separat trainiert und ausgewertet. Zusätzlich wird evaluiert wie gut das trainierte Netz sich verallgemeinern lässt.

### 3.2.1 Laborkamera 1

In diesem Experiment wurde ausschließlich die Ballerkennung auf Daten die von der Laborkamera 1 aufgenommen wurden trainiert. Das Trainingsset enthält dabei 3689 Bilder und das Testset enthält 3688 Bilder.

| | #Bilder | $AP_{25}$ | $AP_{50}$ | $AP_{75}$ | $AP_{95}$ |
|---|---|---|---|---|---|
| Trainings Set | 3689 | 60.57% | 31.52% | 2.53% | 0.0% |
| Test Set View 1 | 3688 | 61.53% | 27.03% | 1.67% | 0.0% |

Tabelle 3.3: Auswertung der Ballerkennung trainiert auf Daten der Laborkamera 1

### 3.2.2 Laborkamera 2

In diesem Experiment wurde ausschließlich die Ballerkennung auf Daten die von der Laborkamera 2 aufgenommen wurden trainiert. Das Trainingsset enthält dabei 3691 Bilder und das Testset enthält ebenfalls 3691 Bilder.

| | #Bilder | $AP_{25}$ | $AP_{50}$ | $AP_{75}$ | $AP_{95}$ |
|---|---|---|---|---|---|
| Trainings Set | 3691 | 68.38% | 24.54% | 1.1% | 0.0% |
| Test Set View 2 | 3691 | 68.38% | 24.51% | 1.12% | 0.0% |

Tabelle 3.4: Auswertung der Ballerkennung trainiert auf Daten der Labor Kamera 2

[2] `https://pjreddie.com/media/files/darknet53.conv.74`

### 3.2.3 RoboCup 2018

Beim RoboCup 2018 in Montreal wurde jedes Spiel mit einer Kamera am Rand des Spielfelds aufgenommen. Die Sichtweise ist ähnlich der zweiten Sicht aus dem Labor. Zum Training wurde ein Video einer Halbzeit benutzt. Dieses Video umfasst 23.214 Frames. Als Testdaten wurde der Video der 2. Halbzeit genutzt mit insgesamt 23.507 Frames.

| | #Bilder | $AP_{25}$ | $AP_{50}$ | $AP_{75}$ | $AP_{95}$ |
|---|---|---|---|---|---|
| Trainings Set | 23.214 | 71.44% | 93.01% | 19.13% | 1.17% |
| Test Set RC18 | 23.507 | 69.32% | 90.7% | 13.08% | 1.01% |

Tabelle 3.5: Auswertung der Ballerkennung im Spiel gegen HTWK in Montreal

## 3.3 Nao Roboter Erkennung

Die Erkennung des Nao Roboters wird erst auf den jeweiligen separaten Datensätzen trainiert und evaluiert und anschließend auf einem gemeinsamen Datensatz getestet.

### 3.3.1 Labor Kamera 1

| | #Bilder | $AP_{25}$ | $AP_{50}$ | $AP_{75}$ | $AP_{95}$ |
|---|---|---|---|---|---|
| Trainings Set | 3691 | 18.3% | 58.9% | 39.9% | 0.0% |
| Test Set View 2 | 3691 | 13.9% | 52.6% | 36.6% | 0.0% |

Tabelle 3.6: Auswertung der Robotererkennung trainiert auf Daten der Labor Kamera 1

### 3.3.2 Labor Kamera 2

| | #Bilder | $AP_{25}$ | $AP_{50}$ | $AP_{75}$ | $AP_{95}$ |
|---|---|---|---|---|---|
| Trainings Set | 3691 | 23.1% | 62.0% | 45.0% | 0.0% |
| Test Set View 2 | 3691 | 21.6% | 61.3% | 42.4% | 0.0% |

Tabelle 3.7: Auswertung der Robotererkennung trainiert auf Daten der Labor Kamera 2

### 3.3.3 RoboCup 2018

Für dieses Experiment werden die selben Trainings- und Testdaten genutzt wie im Ballerkennungsexperiment auch.

| | #Bilder | $AP_{25}$ | $AP_{50}$ | $AP_{75}$ | $AP_{95}$ |
|---|---|---|---|---|---|
| Trainings Set | 23.214 | 34.0% | 72.0% | 58.7% | 0.9% |
| Test Set RC18 | 23.507 | 35.2% | 70.6% | 55.7% | 0.73% |

Tabelle 3.8: Auswertung der Robotererkennung im Spiel gegen HTWK in Montreal.

| **Layer** | Type | Input | Filters | Size/Stride | Output |
|---|---|---|---|---|---|
| **00** | Convolutional | 416 x 416 x 3 | 32 | 3 x 3 | 416 x 416 x 32 |
| **01** | Maxpool | 416 x 416 x 32 | | 2 x 2, stride=2 | 208 x 208 x 32 |
| **02** | Convolutional | 208 x 208 x 32 | 64 | 3 x 3 | 208 x 208 x 64 |
| **03** | Maxpool | 208 x 208 x 64 | | 2 x 2, stride=2 | 104 x 104 x 64 |
| **04** | Convolutional | 104 x 104 x 64 | 128 | 3 x 3 | 104 x 104 x 128 |
| **05** | Convolutional | 104 x 104 x 128 | 64 | 1 x 1 | 104 x 104 x 64 |
| **06** | Convolutional | 104 x 104 x 64 | 128 | 3 x 3 | 104 x 104 x 128 |
| **07** | Maxpool | 104 x 104 x 128 | | 2 x 2, stride=2 | 52 x 52 x 128 |
| **08** | Convolutional | 52 x 52 x 128 | 256 | 3 x 3 | 52 x 52 x 256 |
| **09** | Convolutional | 52 x 52 x 256 | 128 | 1 x 1 | 52 x 52 x 128 |
| **10** | Convolutional | 52 x 52 x 128 | 256 | 3 x 3 | 52 x 52 x 256 |
| **11** | Maxpool | 52 x 52 x 256 | | 2 x 2, stride=2 | 26 x 26 x 256 |
| **12** | Convolutional | 26 x 26 x 256 | 512 | 3 x 3 | 26 x 26 x 512 |
| **13** | Convolutional | 26 x 26 x 512 | 256 | 1 x 1 | 26 x 26 x 256 |
| **14** | Convolutional | 26 x 26 x 256 | 512 | 3 x 3 | 26 x 26 x 512 |
| **15** | Convolutional | 26 x 26 x 512 | 256 | 1 x 1 | 26 x 26 x 256 |
| **16** | Convolutional | 26 x 26 x 256 | 512 | 3 x 3 | 26 x 26 x 512 |
| **17** | Maxpool | 26 x 26 x 512 | | 2 x 2, stride=2 | 13 x 13 x 512 |
| **18** | Convolutional | 13 x 13 x 512 | 1024 | 3 x 3 | 13 x 13 x 1024 |
| **19** | Convolutional | 13 x 13 x 1024 | 512 | 1 x 1 | 13 x 13 x 512 |
| **20** | Convolutional | 13 x 13 x 512 | 1024 | 3 x 3 | 13 x 13 x 1024 |
| **21** | Convolutional | 13 x 13 x 1024 | 512 | 1 x 1 | 13 x 13 x 512 |
| **22** | Convolutional | 13 x 13 x 512 | 1024 | 3 x 3 | 13 x 13 x 1024 |
| **23** | Convolutional | 13 x 13 x 1024 | 1024 | 3 x 3 | 13 x 13 x 1024 |
| **24** | Convolutional | 13 x 13 x 1024 | 1024 | 3 x 3 | 13 x 13 x 1024 |
| **25** | Route 16 | | | | |
| **26** | Convolutional | 26 x 26 x 512 | 64 | 1 x 1 | 26 x 26 x 64 |
| **27** | reorg | 26 x 26 x 64 | | stride=2 | 13 x 13 x 256 |
| **28** | Route 27 24 | | | | |
| **29** | Convolutional | 13 x 13 x 1280 | 1024 | 3 x 3 | 13 x 13 x 1024 |
| **30** | Convolutional | 13 x 13 x 1024 | 30 | 1 x 1 | 13 x 13 x 30 |

Tabelle 3.2: Die Yolov2 Netzwerk Architektur die für die Ball und Nao erkennung genutzt wurde.

# 4 Roboter Lokalisierung

Um die im Bild erkannten Objekte auf dem Feld zu lokalisieren wird als erstes die Kameraposition relativ zum Feldmittelpunkt lokalisiert. Die dafür notwendigen Schritte sind: Feldlinienerkennung, Verzeichniskorrektur der Linse, und die Bestimmung der extrinsischen Kameraparameter.

## 4.1 Feldlinienkennung

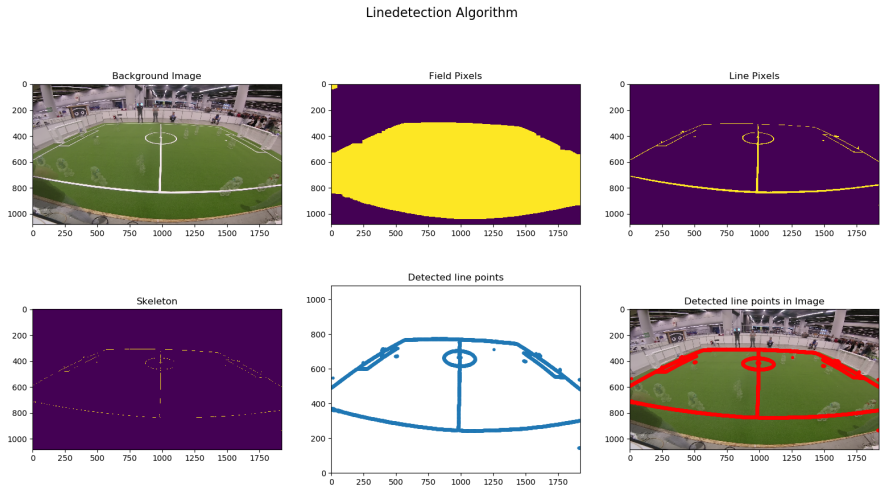



Abbildung 4.1: Darstellung der Ergebnisse der einzelnen notwendigen Schritte zur Feldlinienerkennung.

Im ersten Schritt wird eine Hintergrund Subtraktion auf 30 Bildern in 100 Frame Abständen durchgeführt. Als Background Subtraction Algorithmus wird der BackgroundSubtractorMOG2 aus der OpenCV Bibliothek genutzt. Diese Implementation basiert auf [17] und [18].

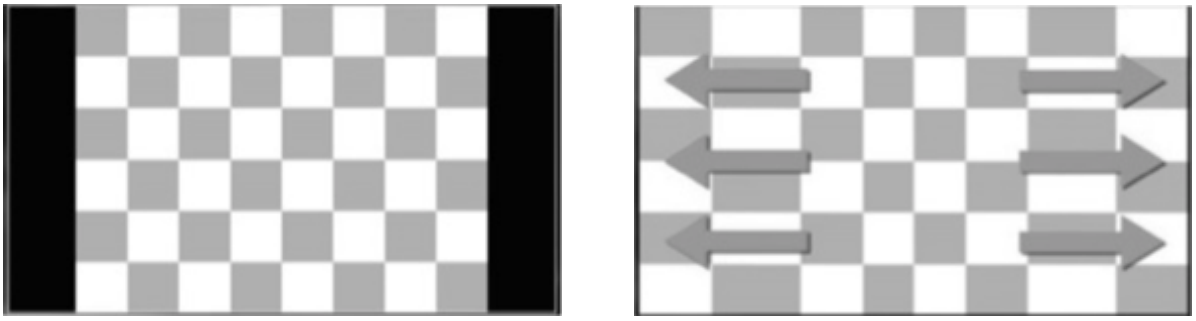
Als nächstes wird das Feld erkannt. Hierfür sind zwei Schwellwerte für grün im HSV Farbraum verwendet. Alle Pixel die zwischen diesen Schwellwerten liegen werden zum Feld

gezählt. Um Rauschen zu unterdrücken werden eine Reihe von morphologischen Operationen angewandt. Die Anzahl und Art der Operationen wurde experimentell bestimmt. Die Pixel, die im Bild als Feldpixel erkannt wurden, sind in Abbildung 4.1 zweites Bild oben gelb dargestellt. Die weißen Linien werden auf ähnliche Art erkannt. Es werden wieder zwei Schwellwerte genutzt und alle Pixel zwischen diesen Werten als weiß erkannt. Die Linienerkennung basiert auf der Annahme das diese die einzigen weißen Pixel im Hintergrundbild innerhalb des Feldes sind. Unter dieser Annahme können die Feldmasken und Linienmasken bitweise verundet werden um die Linien Pixel zu erhalten. Das Ergebnis ist in Abbildung 2.3 oben rechts dargestellt.

Abschließend wird ein *morphological skeleton* Algorithmus auf der Maske angewandt. Dies führt dazu das die Linien im Bild gleich groß sind. Das Ergebnis dieses Algorithmus ist in Abbildung 4.1 links unten zu sehen.

## 4.2 Bestimmung der Position der Kamera

Die Kameraposition kann bestimmt werden indem die Tranformationsmatrix, welche die Linienpunkte im Bild zu den Linienpunkten auf dem Feld projiziert, bestimmt wird. In der aktuellen Implementierung wird eine ungefähre Schätzung der Kameraposition vorausgesetzt. Zum Beispiel wurde die Anfangsschätzung für Aufnahmen beim RoboCup 2018 für die Translation auf (0, -3500, 1800) und die Rotation auf (0, -35, 0) gesetzt.



(a) (b)

Abbildung 4.2: Illustration der GoPro Superview Funktionalitä. Links ist das Originalbild in der nativen 4:3 Auflösung. Rechts ist das Bild nach dem Superview eingestellt wurde.

Die GoPro hat die Besonderheit das sie in einem sogenannten Superview Format aufnehmen kann. Wenn dieser benutzt wird, wird das Bild von dem nativen 4:3 Format auf ein 16:9 Format gedehnt. Dies wird aus ästhetischen Gründen gemacht. In Abbildung 4.2

ist dieses Verfahren visuell dargestellt. Diese Abbildungen sind von der GoPro Hilfe Seite [1] entnommen. Wie die Dehnung genau passiert ist nicht veröffentlicht. Die Streckungsparameter wurden deswegen experimentell bestimmt. In Abbildung 4.3 ist ein vorher-nachher Vergleich für ein Beispielbild dargestellt.

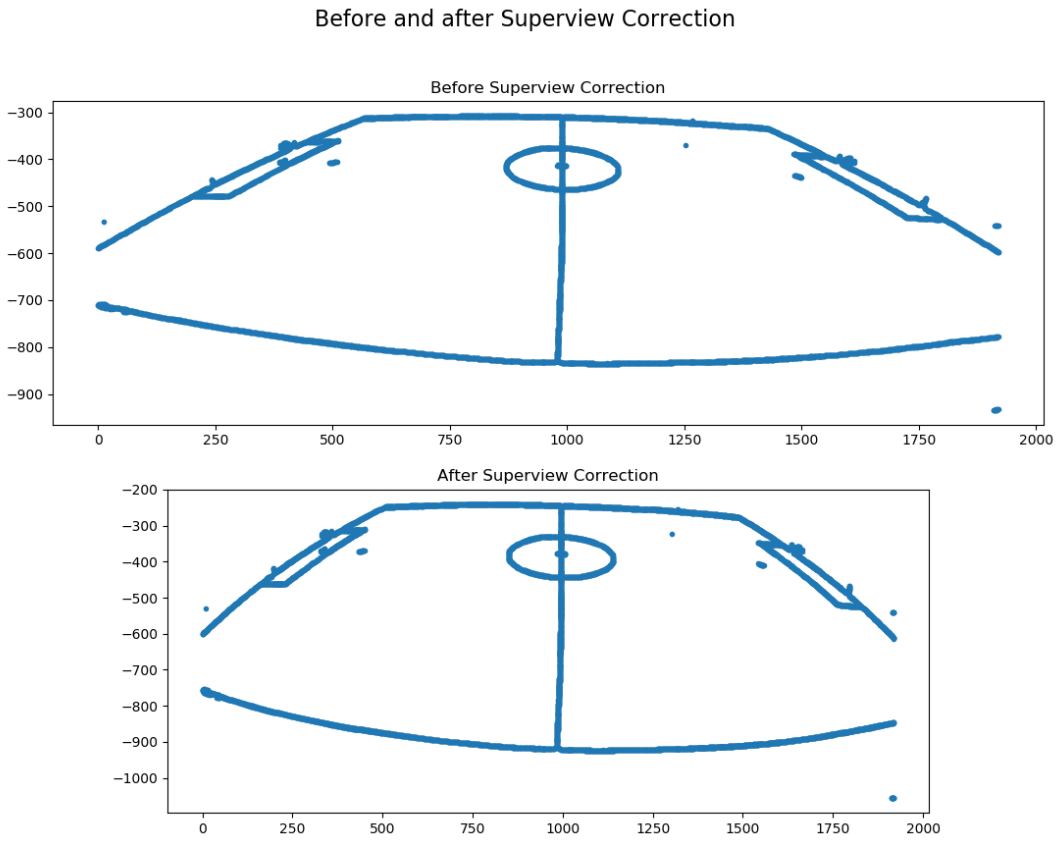



Abbildung 4.3: Vor- und Nachher Bild für die Superview Anpassung.

Es wird ein einfaches radiales Verzeichnungsmodel angenommen. Ferner werden auch nur die 3 radiale Verzeichnungs-Koeffizienten berücksichtigt.

$$x_{distorted} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{4.1}$$

$$y_{distorted} = x(1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) \tag{4.2}$$

Die Koeffizienten $k_1$, $k_2$ und $k_3$ wurden mit der Matlab Toolbox für Kamera Kalibrierung bestimmt.

[1] `https://de.gopro.com/help/articles/question_answer/What-is-SuperView`

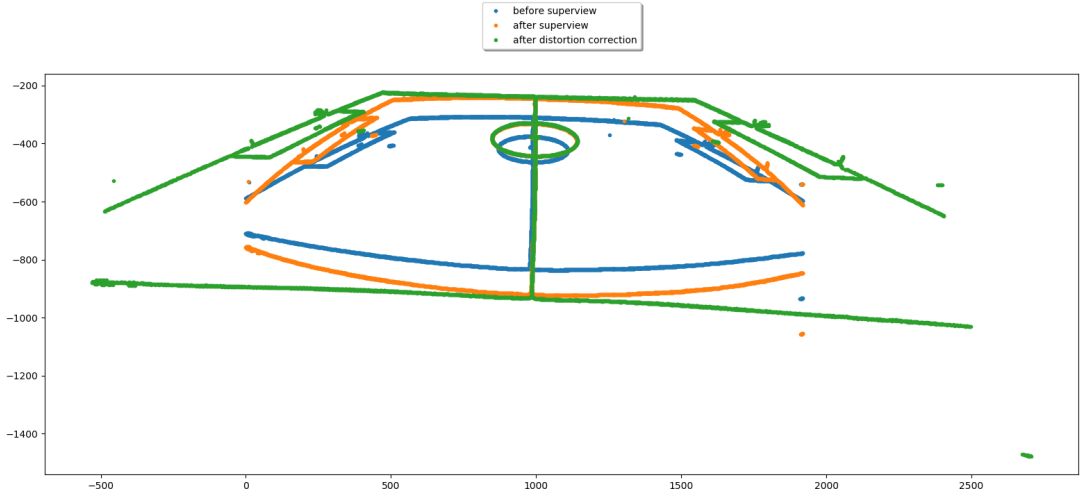



Abbildung 4.4

## 4.3 Lokalisierung der Roboter

Es wurden zwei Experimente durchgeführt um die Genauigkeit der Roboterlokalisierung zu evaluieren. Im ersten Experiment dribbelt ein Roboter einen Ball über das Feld in Richtung des gegnerischen Tores.Der Roboter ist dabei alleine auf dem Feld. Der Roboter wird zum einen von dem vorgestellten System verfolgt sowie durch ein extra Aufnahmesystem OptiTrack[2]. Das Optritrack System besteht aus 10 Infrarot Kameras die rundum das Spielfeld angebracht sind. Dieses System ist in der Lage an dem Roboter angebrachte reflektierende Marker mit einer Genauigkeit von unter 1mm und einer Framerate von 120 FPS zu verfolgen. Die Marker wurden in diesem Experiment am Kopf des Roboters angebracht. Die Marker sind in Abbildung 4.5 (links) zu sehen.

In diesem Experiment wurde die Kamera mit einem mittleren Fehler von 66mm der projizierten Linienpunkte lokalisiert. Diese Kameraposition wurde genutzt um die die mit YOLO erkannten Roboter Positionen auf das Feld zu projizieren. Für die Projektion wurde der Mittelpunkt der Bounding Box als Approximation des Mittelpunktes des Roboterkörpers benutzt. Bei der Projektion wurde angenommen das dieser Mittelpunkt eine Höhe von 260mm hat. Der daraus resultierende Pfad des Roboters ist in Abbildung 4.5 (rechts) dargestellt. Der mittlere Fehler zwischen den Optitrack Pfad und dem aus dem Video berechneten Pfad ist 14,93mm mit einer Standardabweichung von 98mm. Für dieses recht simple System ist diese Genauigkeit überraschend genau.

Für das zweite Experiment wurde eine Aufnahme des Viertelfinales des RoboCup Meisterschaft 2018 genutzt. In diesem Spiel hat das Team HTWK Leipzig gegen Berlin United

[2]https://optitrack.com/

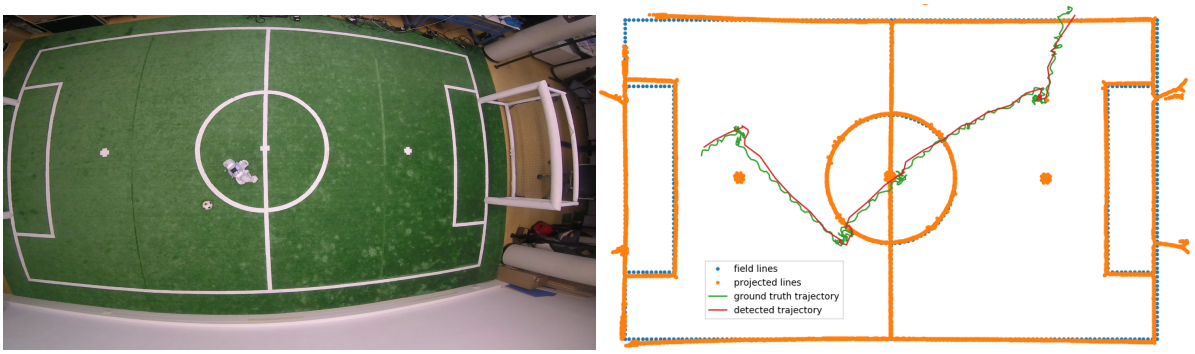



Abbildung 4.5: Lokalisierung eines Roboters unter Laborbedingungen. Links: Ausschnitt aus der Videoaufnahme. Rechts: Zurückgelegter Pfad des Roboters berechnet aus den Videodaten und die aufgenommenen Groundtruth aus dem Optitrack System. Die projizierten Linienpunkte sind ebenfalls dargestellt zusammen mit dem Feldmodel. Die Torpfosten werden hier als Linien behandelt.

gespielt. Die Kamera konnte wieder mit einem mittleren Fehler von 66mm basierend auf den projizierten Linienpunkten lokalisiert werden. Abbildung 4.6 (rechts) zeigt die Pfade der Roboter während der READY Phase. Das sind die ersten 45 Sekunden des Spiels in denen die Roboter auf ihre Startpositionen laufen. Die erkannten Positionen sind in Grün dargestellt. Die Positionen wie sie die Roboter selber berechnet haben sind in Rot abgebildet. Diese Informationen wurden aus der Netzwerk Kommunikation extrahiert.

Der mittlere Abstand der beiden Pfade ist für die Roboter des HTWK Teams 87mm mit einer Standard Abweichung von 82mm. Für die Roboter des Berliner Teams ist der mittlere Abstand 55mm mit einer Standardabweichung von 44mm. Der größere Abstand bei HTWK liegt vorallem an der ungenaueren Positionsschätzung der HTWK Roboter. Dies ist in Abbildung 4.6 (links) gut zu sehen.

Die Ergebnisse der beiden Experimente zeigen das eine visuelle Verfolgung der Roboter möglich ist mit einem simplen Kameraaufbau ohne spezielle Hardware. Zusammen mit der kommunizierten Positionsdaten ist es mögliche die Roboter Pfade zu rekonstruieren. Dies kann beispielsweise für Verifikation der Roboter Lokalisierung oder Teamverhaltensanalysen genutzt werden.

In der aktuellen Implementierung beinhaltet der Ansatz noch eine Reihe von Heuristiken und Vereinfachungen die zum Lokalisierungsfehler beitragen. Vorallem führt die Annahme der fixen Roboterhöhe zu Fehler in der Positionsbestimmung. Die Höhe der Roboter kann sich je nach Laufverhalten ändern und vor hingefallene Roboter stimmt diese Annahme überhaupt nicht.

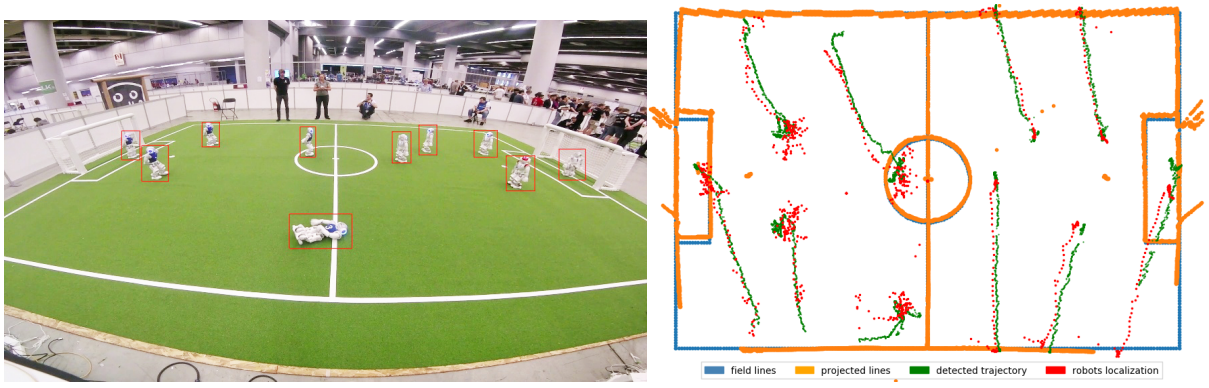



Abbildung 4.6: Lokalisierung der Roboter unter normalen Feldbedingungen während der *READY* Phase. Links: Endgültige Positionen der Roboter zum Ende der READY Phase. Rechts: aus den Videodaten berechnete Positionen der Roboter [Grün] und kommunizierte Positionen der Roboter in [Rot]. Die projizierten Linienpunkte sind ebenfalls dargestellt zusammen mit dem Feldmodel. Die Torpfosten werden hier als Linien behandelt.

Die Erkennung der Spielernummer eines Roboters ist noch nicht implementiert. Eine Möglichkeit dies zu zu implementieren ist es die Zuordnung der Startpositionen zu Spielernummern einmal vor dem Spiel festzulegen und diese über den Verlauf des Spiels zu tracken. Die berechneten Positionen der Roboter welche kommuniziert werden können ebenfalls zur Bestimmung der Spielernummer benutzt werden.

# 5 Fazit

Diese Arbeit hat die Kamera Lokalisierung sowie Ball und Robotererkennung und deren Lokalisierung auf dem Feld vorgestellt. Erste Ergebnisse zeigen das ein genaues Verfolgen der Roboter und des Balls aus einer Kamerasicht möglich ist. Allerdings gibt es noch Verbesserungspotential in allen Aspekten des Aufnahmesystems.

Die Felderkennung basiert stark auf Farben. Auch die Anzahl der morphologischen Operationen ist heuristisch für die Videodaten dieser Arbeit bestimmt wurden. Der aktuelle Ansatz sollte durch eine robuste Felderkennung die auch unter unterschiedlichen Lichtbedingungen klarkommt ersetzt werden. Die Felderkennungsalgorithmen die auf dem Nao eingesetzt werden können beispielsweise als Inspiration dienen.

Damit die vorgestellte Kameralokalisierung klappt muss bereits eine grobe Positionsschätzung vorliegen. Dies verhindert zurzeit dass das Aufnahmesystem ab beliebiger Stelle aufgestellt werden kann. Eine Mögliche Alternative zum aktuell implementierten Ansatz ist in [6] zu finden.

Die Robotererkennung mit dem YOLO Algorithmus ist schon sehr genau. Die Ballerkennung hingegen hat nur eine unzureichende Genauigkeit. Dies liegt daran das hier die unveränderte YOLO Architektur benutzt wurde welche Probleme hat kleine Objekte zu erkennen. Seit der Publikation des YOLO Papers wurden eine ganze Reihe von Deep Learning Architekturen vorgestellt die dieses Problem angehen und zusätzliche eine bessere Inferenz Zeit und Genauigkeit versprechen. Ein weitere Möglichkeit die Präzision der Erkennung zu verbessern ist mehr Trainingsdaten zu verwenden. Um den zeitlichen Aufwand der Annotation möglichst gering zu halten können die Netze die in dieser Arbeit trainiert wurden zum vor labeln genutzt werden. Labeltools wie CVAT unterstützen bereits eine Integration von Tensorflow Modellen für genau diese Aufgabe.

Um die Lokalisierung zu verbessern braucht man eine bessere Abschätzung wo sich der Fußpunkt des Roboters in der Bounding Box befindet. Eventuell reicht es den Mittelpunkt der unteren Kante zu benutzen. Eine weitere Möglichkeit wäre statt einer Objekt Detektion eine Semantische Segmentierung durchzuführen. Dies würde aber eine bedeutend hohen Mehraufwand beim labeln der Trainingsdaten bedeuten.

In dieser Arbeit wurde eine Roboter- und Ballerkennung auf Basis von YOLO vorgestellt und aufbauend darauf eine Objektlokalisierung auf dem Feld. Möglichkeiten zur Verbesserung der einzelnen Elemente des Systems wurden besprochen.

# Literaturverzeichnis


[1] `https://github.com/pjreddie/darknet`.

[2] `https://github.com/AlexeyAB`.

[3] `https://github.com/qqwweee/keras-yolo3`.

[4] `https://github.com/YunYang1994/tensorflow-yolov3`.

[5] Panna Felsen, Patrick Lucey, and Sujoy Ganguly. Where Will They Go? Predicting Fine-Grained Adversarial Multi-agent Motion Using Conditional Variational Autoencoders. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11215 LNCS:761–776, 2018.

[6] Namdar Homayounfar, Sanja Fidler, and Raquel Urtasun. Soccer Field Localization from a Single Image. pages 1–16, 2016.

[7] Hoang M. Le, Carr Peter, and Yisong Yue. Data-Driven Ghosting using Deep Imitation Learning. *MIT Sloan Sports Analytics Conference*, pages 1–15, 2017.

[8] Heinrich Mellmann, Stella Alice Schlotter, and Christian Blum. Simulation Based Selection of Actions for a Humanoid Soccer-Robot. In *RoboCup 2016: Robot Soccer World Cup XX*, 2016. to appear.

[9] Heinrich Mellmann, Stella Alice Schlotter, and Philipp Strobel. Toward data driven development in robocup. In *RoboCup 2019: Robot Soccer World Cup XXIII*, 2019. to appear.

[10] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[11] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[12] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[13] Stella Alice Schlotter. Selection of Actions based on Forward Simulations. Bachelorarbeit, Humboldt-Universität zu Berlin, 2017.

[14] Lucas Schwaß. Visuelle Verfolgung humanoider Roboter im RoboCup-Kontext. Diplomarbeit, Humboldt-Universität zu Berlin, Oktober 2017.

[15] Philipp Strobel. Methoden und Werkzeuge zur Analyse von Teamverhalten und -strategie im Roboterfußball. Studienarbeit, Humboldt-Universität zu Berlin, 2018.

[16] Yisong Yue, Patrick Lucey, Peter Carr, Alina Bialkowski, and Iain Matthews. Learning Fine-Grained Spatial Models for Dynamic Sports Play Prediction. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2015-Janua(January):670–679, 2014.

[17] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31 Vol.2, Aug 2004.

[18] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773 – 780, 2006.