Positionserkennung im dreidimensionalen Raum mittels zwei dimensionaler Daten am Beispiel eines Tennisballsammel-Roboters

KI und Big Data

Projektarbeit Sommersemester 2025

Michael Kaup

s0585949

Contents

1. Einführung

Die vorliegende Belegarbeit befasst sich mit dem Thema der Positionserkennung mittels Machine Learning. Genauer soll es darum gehen dreidimensionale Positionsdaten aus zweidimensionalen Daten zu bestimmen. Diese zweidimensionalen Daten liegen als Bilddaten vor. Der Anwendungsfall ist ein mobiler Roboter, der einen oder mehrere Tennisbälle auf einem Tennisplatz einsammeln soll. Es soll herausgearbeitet werden, ob es möglich ist, ein Machine learning Model darauf zu trainieren, präzise die Position eines Tennisballs auf dem Tennisfeld anhand eines Bildes zu bestimmen.
Die Arbeit gliedert sich in mehrere Abschnitte. Zuerst wird die Problemstellung erläutert. Anschließend wird die Beschaffung der notwendigen Daten beschrieben. Auf diesen wird folgend eine Analyse durchgeführt sowie Feature Engineering betrieben. Im nächsten Schritt wird das Modelltraining erläutert, mit abschließender Evaluierung.

2. Problemstellung

Diese Arbeit basiert auf der Idee, einen Roboter zu entwicklen, der in der Lage ist, Tennisbälle auf einem Tennisplatz zu erkennen und einzusammeln. Der Roboter soll autonom agieren und in der Lage sein Tennisbälle zu erkennen, deren Position zu bestimmen, über den Tennisplatz navigieren, Hindernisse erkennen und vermeiden, um dann die Tennisbälle einsammeln und an einem bestimmten Ort ablegen zu können.
Die Herausforderung besteht darin, dass der Roboter ohne weitere Hilfsmittel, wie z.B. eine externe, den Platz überblickenden Kamera die Positionen der Tennisbälle bestimmen soll. Hierfür soll der Roboter mit mindestens einer Kamera ausgestattet werden, die Bilder des Tennisplatzes aufnimmt. Diese Bilder sollen dann mit Hilfe eines trainierten Machine Learning Modells analysiert werden, um die Position der Tennisbälle zu bestimmen. Erste Überlegungen zur Form des Roboters lassen sich mit der eines Staubsaugerroboters vergleichen, der mit zwei Rädern angetrieben wird. Das erschwert die Aufgabe der Positionsbestimmung insofern, dass der Roboter relativ klein ist und die Kamera in einem relativ flachen Winkel auf den Tennisplatz gerichtet ist.
Der Roboter existiert noch nicht, er ist nur ein theoretisches Konstrukt.

3. Zielstellung

Diese Arbeit dient dazu erste Schritte in Richtung der finalen Positionserkennung mittels Bilddaten zu gehen. Es soll ein erster Ansatz für die Positionsbestimmung im drei dimensionalen Raum anhand zweidimensionaler Bilddaten geprüft werden und dessen Machbarkeit beurteilt werden. Dafür werde ich mich auf eine Kombination aus Objekterkennung und Regression stützen, in der ein Tennisball erkannt wird und dessen Entfernung zur Kamera/zum Roboter vorhergesagt werden soll.

4. Datenbeschaffung

4.1. Anforderungen an die Daten

Für die vorliegende Arbeit wird ein Datensatz mit sog. unstrukturierten Daten benötigt. Diese Daten sollen in der Form von Bildern vorliegen. Die Bilder sollen Tennisbälle auf einem Tennisplatz zeigen und aus der Höhe eines mobilen, autonomen, Staubsaugerroboter ähnlichen Roboters aufgenommen werden.

TODO: ausformulieren < Desweiteren:

- verschiedene Höhen?
- verschiedene Winkel
- verschiedene Abstände?
- verschiedene Hintergründe?
- mit und Ohne Menschen?

WEG?: Die Daten können Synthetisch oder Reale Daten sein sowie auf jeweils auf verschiedene Arten und Weisen beschafft werden. Im folgenden werden beide Ansätze und ihre Beschaffungsmethoden diskutiert. >

4.2. Reale Daten

Reale Daten sind Daten, die echte Ereignisse, Zustände oder Interaktionen abbilden. Diese Daten werden meist durch Beobachtung gewonnen. Reale Daten können selber erzeugt oder aus externen Quellen beschafft werden.

4.2.1. Reale Daten selber erzeugen

Daten können selber erzeugt werden. Im falle unstrukturierter Bilddaten hieße das das Aufnehmen von Bildern mit einer Kamera. In unserem Anwendungsfall des Roboters, könnte diese Kamera z.B. auf dem Roboter montiert sein, der dann Bilder von Tennisbällen auf einem Tennispaltz aufnimmt. Dies würde reale Daten erzeugen, die sehr nah an der Problemstellung leigen. Da es sich bei dem Roboter um ein theoretisches Konstrukt handelt, ist es nicht möglich, die Daten mit einem echten Roboter zu beschaffen. Um trotzdem möglichst realistische Daten zu erhalten ist es alternativ möglich, mit einer von einem Menschen gehaltenen Kamera Bilder auf einem Tennisplatz aufzunhemen. Dafür könnten Tennisbälle an verschiedenen Positionen auf dem Tennisplat platziert und aus verschiedenen Winkeln und Entfernungen fotografiert werden. Dabei wäre es wichtig zu beachten, zuvor eine feste Höhe zu bestimmen, auf der sich die Kamera des Roboters später einmal befinden soll. Die Kamera mit der die Bilder zum Training des Modells dann aufgenommen werden, sollte dann für jedes Foto auf dieser Höhe platziert werden um für den späteren Anwendungsfall relevante Daten zu erhalten. Es sollte zusätzlich in betracht gezogen werden, die Höhe bei jedem Bild, innerhalb eines Tolleranzbereichs, zu variieren oder für jede Kombination von Aufnahmewinkel und -position mehrere Bilder um die bestimmte Höhe herum aufzunehmen. Dies könnte ausschließen, dass das Modell nur auf einer bestimmten Höhe zuverlässig funktioniert.
TODO: ausformulieren <
Vorteile:

- realistische Daten
- entsprechen genau dem späteren Anwendungsfall

Nachteile:

- teuer, da Tennisplatz und Tennisbälle gemietet werden müssen
- nicht reproduzierbar
- Zeitaufwändig für große Datenmengen

>

4.2.2. Internetrechere

Es ist möglich realistische Daten zu erhalten, ohne diese selber gesammelt zu haben. Plattformen wie kaggle oder huggingface bieten neben vortrainierten Machine Learning Modellen öffentlich zugängliche Datensätze an, die genutzt werden können um eigene Modelle zu trainieren. TODO: ausformuleiren Vorteile:

- realistische Daten
- weniger Zeitaufwand

Nachteile:

- entspricht nicht genau dem späteren Anwendungsfall
- nicht für alles existiert bereits ein Datensatz

>

4.2.3. Konklusion, dass ich nicht realistische Daten nutzen werde

Auch wenn die Nutzung realer Daten viele Vorteile bietet, wie z.B. die mögliche Nähe zum späteren Anwendungsfall, ist es für mich aufgrund der genannten Nachteile nicht attraktiv reale Daten für diese Projektarbeit zu nutzen. Vor allem der Zeit- und Kostenaufwand bei der manuellen Beschaffung, sowie die spezifizität der Anforderungen an den Datensatz, die es schwer machen entsprechend relevante Daten für meinen Anwendungsfall mittels Internetrecherche zu finden, sind schwerwiegende Argumente gegen eine Nutzung realer Daten.

4.3. Datenbeschaffung: Vertiefung

4.3.1. Synthethische Daten

Synthetische Daten werden im Gegenstz zu realen Daten nicht durch beobachtungen gewonnen, sondern erzeugt. Sie werden künstlich generiert und können einen realen Datensatz entweder ersetzen (Vollsynthethisch), teilweise ersätzen (Teilsynthetisch) oder ergänzen (Hybrid). Für meinen Anwendungsfall werde ich ausschließlich künstlich generierte Daten verwenden, da es keine für diesen spezifischen Anwendungsfall bereits existierenden Daten gibt.

4.3.2. Diskussion der beiden synthetischen Ansätze

Zum Zeitpunkt dieser Arbeit habe ich zwei Ansätze zur Erzeugung der Daten in Betracht gezogen. Der erste ist die automatische Erzeugung der Daten mittels eines Bilderzeugungs Models. Der Zweite die teilweise manuelle Erzeugung der Daten mittels einer Szene in Unity. Diese Ansätze werden im folgenden Erläutert, getestete und diskutiert.

4.3.2.1. Image Generation

Das erzeugen künstlich generierter Bilder ist heute einfacher denn je. Viele generative Bildmodelle stehen heute im Internet zur Verfügung. Der Vorteil der generierung künstlicher Bilder durch ein generatives Bildmodel ist, das theoretisch eine große Menge an Bildern in kurzer Zeit erezugt werden kann. In meinem Fall habe ich testweise die Bildgenerierungsfunktionen der beiden Large Language Models LeChat von Mistral AI und ChatGPT von OpenAI verwendet. LeChat verwendet zur Erzeugung von Bildern das FLUX.1 Diffusionsmodell von BlackForest Labs. ChatGPT verwendet das Diffusionsmodell DALL-E 3 von OpenAI.

4.3.2.1.1. Durchführung

Ich habe für die Erzeugung der Bilder folgenden Prompt benutzt:

"Erzeuge ein Bild eines Tennisballs auf einem Tennisplatz. Der Tennisplatz ist in einer Sporthalle. Es sollen keine Menschen auf dem Bild erkennbar sein. Das Bild soll aus ca. 20cm Höhe aufgenommen worden sein. Der Ball soll einige Meter weit von der Kamera entfernt sein. Wähle diese Entfernung beliebig. Bitte schreibe, nachdem du das Bild erzeugt hast, wie weit der Tennisballauf dem Bild wirklich von der Kamera entfernt ist."

Es wurden mit jedem Modell mehrere Bilder erzeugt, sofern das mit den kostenlos zugänglichen Versionen der beiden LLMs möglich war. LeChat erzeugte zuverlässlich Bilder mit einem Tennisball auf einem Tannisplatz, in einer Sporthalle. Jedoch waren die Abstände des Balls zur Kamera immer die gleichen, während die Angabe der Entfernung von LeChat variierte. Zusätzlich lässt eine logische Beurteilung der größe des Balls sowie der stilistischen Art des Bildes darauf schließen, dass die Entfernungsangaben des

LLMs nicht korrekt sind. Der Ball nimmt immer ungefähr ein viertel des Bildes ein und der Unschärfeeffekt suggeriert, dass es sich bei der Aufnahme um eine Nahaufnahme handeln soll. LeChat behauptet jedoch, dass die Bälle drei oder vier Meter entfernt seien sollen.

Folgend sind die Erzeugten Bilder sowie die dazugehörigen Antworten von LeChat zu sehen:

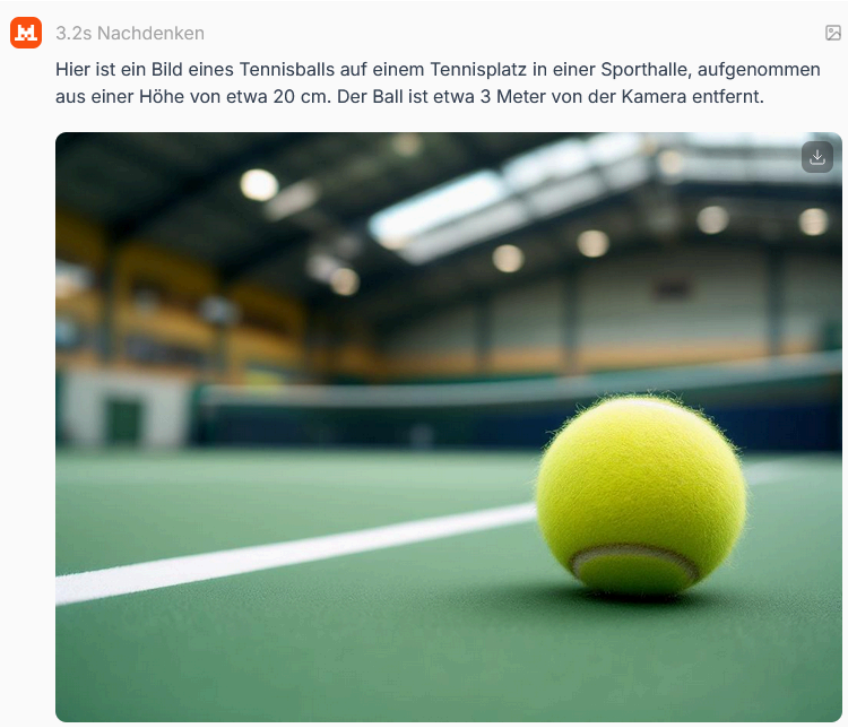

Figure 1: Beschreibung und Bildausgabe von LeChat.

Figure 2: Beschreibung und Bildausgabe von LeChat.

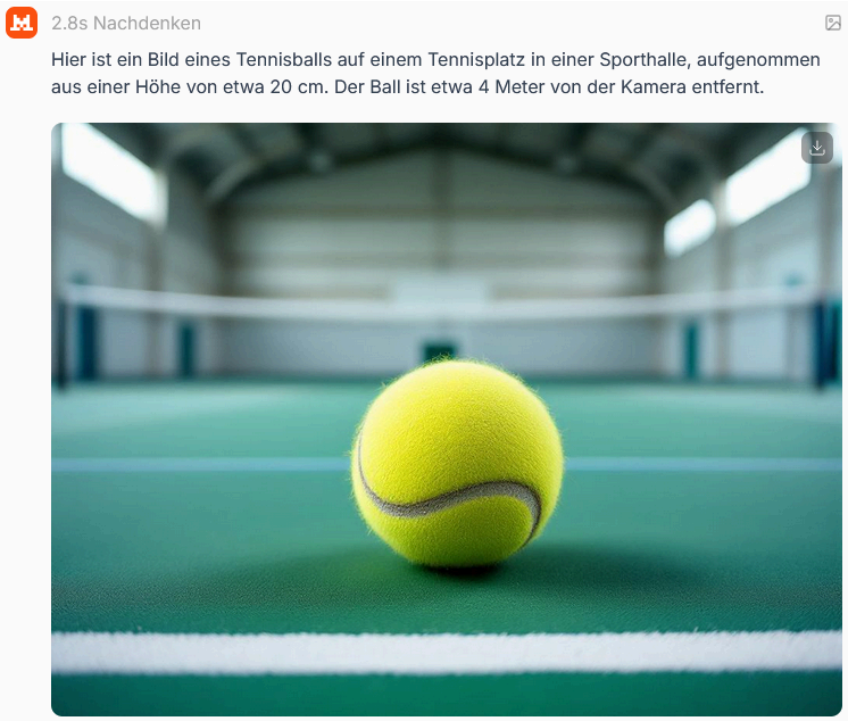

Figure 3: Beschreibung und Bildausgabe von LeChat.

4.3.2.1.2. Bewertung

Aus den erhaltenen Ergebnissen lässt sich schließen, dass generative Bildmodelle nicht dafür geeignet sind, Maßangaben entsprechend ihrer realen Gegebenheiten nachzuah-

men. Insofern wird die Erzeugung der Beispieldaten auf diese Art und Weise nicht weiter in Betracht gezogen.

4.3.2.2. 3D Modelling

Virtuelle, dreidimensionale Umgebungen werden heute bereits vielseitig zum Training verschiedener Anwendungen eingesetzt. Beispielhaft ist hier die Robotik, in der z.B. die von NVIDIA entwickelte Simulatiosnumgebung Isaac Sim nutzt, um virtuellen Klonen ihrer Roboter Bewegungsabläufe beizubringen, die dann auf die reale Welt übetragen und in dieser genutzt werden können. Diese Simulationsumgebungen stellen ebenfalls unstrukturierte, synthetische Daten dar, anhand derer z.B. ein KI/Machine Learning Model für die Bewegung eines Roboters trainiert werden kann. Man kann virtuelle Umgebungen auch für das Erzeugen von Bilddaten verwenden, indem man eine virtuelle Szene dessen erzeugt, was man von einem Model klassifiziert oder erkannt haben möchte. In meinem Anwendungsfall wäre eine solche Szene z.B. ein virtueller Tennisplatz. Auf diesem kann dann ein Virtueller Tennisball platziert werden und ein Bild dieser Szene aufgenommen werden, welches dann für das Training eines Modells zu Erkennung der Entfernung eines Tennisballs vom Roboter genutzt werden kann. Vorteile dieser Vorgehensweise sind, dass der Prozess der Bildaufnahme und Platzierung des Balls relativ Zeit unaufwendig sind und schnell verschiedene Positionen des Balls und der Kamera eigenommen werden können. Desweiteren kann die Entfernung des Balls zum Bild genau bestimmt werden, da es in Modellierungs und Engine Software möglich ist, Objekte Maßstabgetreu einzufügen, zu platzieren und zu skalieren.

4.3.2.2.1. Durchführung

Idee: Modellierung einer Szene in Unity Tennisplatz in einer “Halle”. Halle ist ein Cube. Tennisball auf dem Tennisplatz. Plan: Mit Blender die Maße des Platzes und des Balls messen. Entsprechend sklaieren, dass beide realistische Maße haben. Tennisplatz ca. 23,77 m x 10.97 m [1]. Tennisball Durchmesser ca. 6.5 cm [2]. Ich habe mich für einen Durchmesser von 6.6cm entschieden was in Unity eienr Scale des Balls von 3.4 in alle Richtungen entspricht. Die Kamera wird an einem festen Punkt platziert, in der Mitte de Platzes, direkt neben dem Netz, mit Blick vom Netz weg zum Spielfeldrand. Der Ball

soll dann, in der Theorie, durch ein Script an zufälligen Stellen innerhalb des Blickfeldes der Kamera platziert werden. Dann soll das script einen Screenshot machen und die Entfernung des Balls zur Kamera berechnen. Die Entfernung des Balls wird dann in einem Textdokument gespeichert, das wie der Screenshot heißt.

4.3.3. Versuchsaufbau des gewählten Ansatzes

Kamera: Es wurde versucht die Einstellungen der virtuellen Kamera so realistisch wie möglich an die der evtl. später genutzten RaspBerry Pi Camera Module 3 [3] anzupassen. Einige Einstellungen lassen sich aber nicht 1 zu 1 übernehmen. Z.B. kann das Field of View nur entweder Hotizontal oder Vertikal angepasst werden. Wobei eine Änderung des einen Wertes, den anderen mit verändert. Ich habe mich dann z.B. auf eine horizontale FOV von 66° festgelegt, da ich die breite des aufgenommenen Szenenausschnittes nicht unnötig verbreitern wollte. Entgegen der in der Doku stehenden 41° für die vertikale FOV beträgt die vertikale FOV in Unity dann ca. 34.4°. Ein weitere Wert, der sich nicht ändern lässt, ohne die Einstellungen für die FOV zu beeinflussen ist die Focal Length. Es kann sein, dass ich die Einstellungen nicht vollends verstehe, ich wollte mich aber auch nicht zu lange mit der Kamera aufhalten. Sollte es aber nicht möglich sein, die Werte einer realen Kamera vollständig zu übernehmen, könnte das eventuell einen Einfluss auf die Leistung des Models in einer realen umgebung haben, da unterschiedliche FOV Werte Objekte unterschiedlich darstellen/verzerren, da eine Kamera mit einem größeren FOV dazu neigt, Objekte Objekte kleiner darzustellen, als mit kleineren FOV Werten.

Kamera auf der Y Achse in 20cm Höhe platziert, Annahme, dass das die Höhe der Kamera auf dem Roboter sei. Auf der X-Achse -14.54 und 0 auf der Z-Achse.

Ball: Der Ball soll zufällig im FOV Radius Feld der Kamera platziert werden. Die Höhe Y-Achse bleibt immer gleich auf 0.034. Zuerst wird nur der Z-Achsen-Wert, also die Entfernung verändert. X-Achse bleibt -14.5. Der Ball befindet sich also in einer Linie mit der Kamera auf der X-Achse. Der vorerst minimal einnehmbare Z-Achsen Wert ist 0.5, der maximale Wert ist 6.5. Der Ball soll auf der Y-Achse bei jeder Platzierung um einen zufälligen Wert gedreht werden, was die Platzierung des Balls annähernd realistischer macht.

Ich habe ein C-Sharp Script implementiert, dass beim Starten der Unity Scene im Game Window den Ball auf einer Zufälligen entfernung paltziert und zufällig rotiert. Für jeden Positionswechsel wird ein screenshot der neuen Position aufgenommen und die Distanz zur Kamera berechnet. Dann wird der Screenshot gespeichert und zusammen mit dem Namen des Screenshots, die Distanz des Balls in diesem Screenshot in einer Liste gespeichert. Diese Liste wird, wenn so viele Screenshots wie angegeben erstellt wurden in ein JSON umgewandelt.

Problematiken:

- verschwommene Tennisbälle, weil das neu positionieren des Tennisballs gleichzeitig mit der Aufnahme des Screenshots stattfand. Lösung: einige Frames nach der Transformation abwarten.
- (Unity erstellt für jede Datei eine meta Datei)

Das script habe ich dann an einem Objekt der Unity Szene hinzugefügt, wodurch es beim Start der Szene ausgeführt wird.

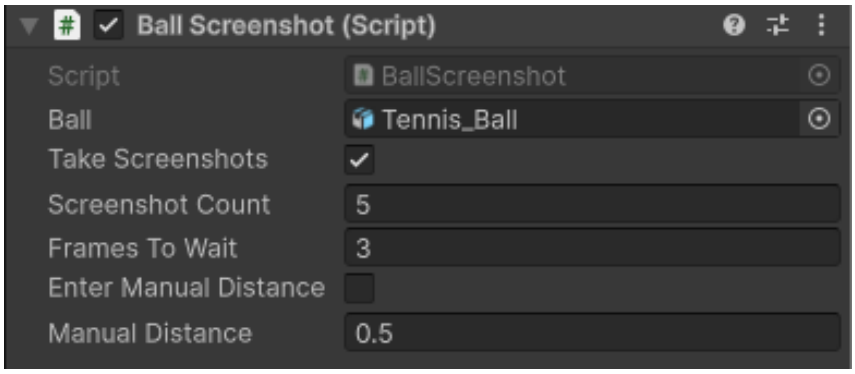

Figure 4: Inputmaske der Scriptparameter für das Aufnehmen der Screenshots in Unity.

Wie in Figure 4 zu sehen können verschiedene Parameter für das Script vor dem Start eingestellt werden. Relevant ist hier jedoch nur, dass die Anzahl der aufzunehmenden Screenshots bestimmt werden kann.

4.3.4. Durchführung des gewählten Ansatzes

Die aufgenommenen Screenshots orientieren sich in ihrer Größe an der eingestellten Auflösung des Gamewindows in Unity. Da die Objekterkennung später mit YOLO geschehen soll, habe ich die Displaygröße auf 640x640 Pixel gestellt. Das ist die empfohlene Auflösung für YOLO inputs.

Dann gebe ich die gewünschte Anzahl an Screenshots im Screenshot Count Parameter ein und starte die Unity Szene.

Die Screenshots werden nach jeder Platzierung des Balls automatisch aufgenommen und zusammen mit den Annotationen für die Distanz in einem Ordner mit dem aktuellen Datum und der aktuellen Uhrzeit als Namenskombiantion gespeichert.

5. Explorative Datenanalyse

Wie bereits erwähnt handelt es sich bei den hier verwendeten Daten um unstrukturierte Bilddaten. Die Anzahl der Daten beträgt so viele wie bei der Ausführung des Scripts angegebene Screenshot Counts. Ein erster Testlauf nachdem das script zufriedenstellend funktionierte enthielt 50 Datensätze. Datentype sind vom Typ .png.

Alle Bilder enthalten wie gewünscht eine Abbildung des Tennisballs in einer beliebeigen Entfernung zur Kamera.

Zusätzlich zu den unstrukturierten Bilddaten erhalten wir außerdem zwei Datensätze an strukturierten Daten. Der eine in Form einer .json Datei, der andere in Form einer .csv Datei. Beide enthalten die Annotationen der Distanz für jede der Bilddateien.

Die JSON Datei enthält eine Liste an Annotationen. Jede Annotation verfügt über den key ImageName, der den Namen der zur Annotation gehörigen Bilddatei enthält, sowie den key Distance, der die Distanz zur Kamera enthält. Die Listeneinträge sind in der Riehenfolge angeordnet in der sie aufgenommen wurden, also von Index i = 0 bis i = n, wobei n die Anzahl der zu Scriptbeginn festgelegten Screenshots ist. In unserem Testfall befinden sich 50 Listenobjekte in der JSON Datei. Der key ImageName enthält ein string-value. Der key Distanz ein float-value mit mehreren Nachkommastellen.

Das gleiche liegt für die csv-Datei vor. Sier verfügt über zwei Spalten, ImageName und Distance. Beide Spalten enthalten die selben Daten wie die JSON Datei.

Die korrektheit der Ausgegebenen Distanzen wurde mehrmals mit der eingabe Manueller Distanzen überprüft. Die Namen der Bilddateien stimmen in beiden Fällen mit denen der entsprechenden Bilddaten überein.

Anhand des eda notebooks lässt sich erkennen dass für die Anzahl n an Bidlern je ein Datenpunk vorliegt.

Ein erster Test, die Bilder mit Yolo annotieren zu lassen verlief wenig erfolgreich. Yolov8n kann Objekte aus bereits 80 vortrainierten Objektlklassen erkennen und annotieren. Eine Davon ist die Sprotsball Klasse, zu der unter anderem Tennisbälle gehören. Aus einer Menge von 50 Bildern erkannte Yolo auf keinem den Tennisball als Sportsball, wie in Figure 5 zu sehen.

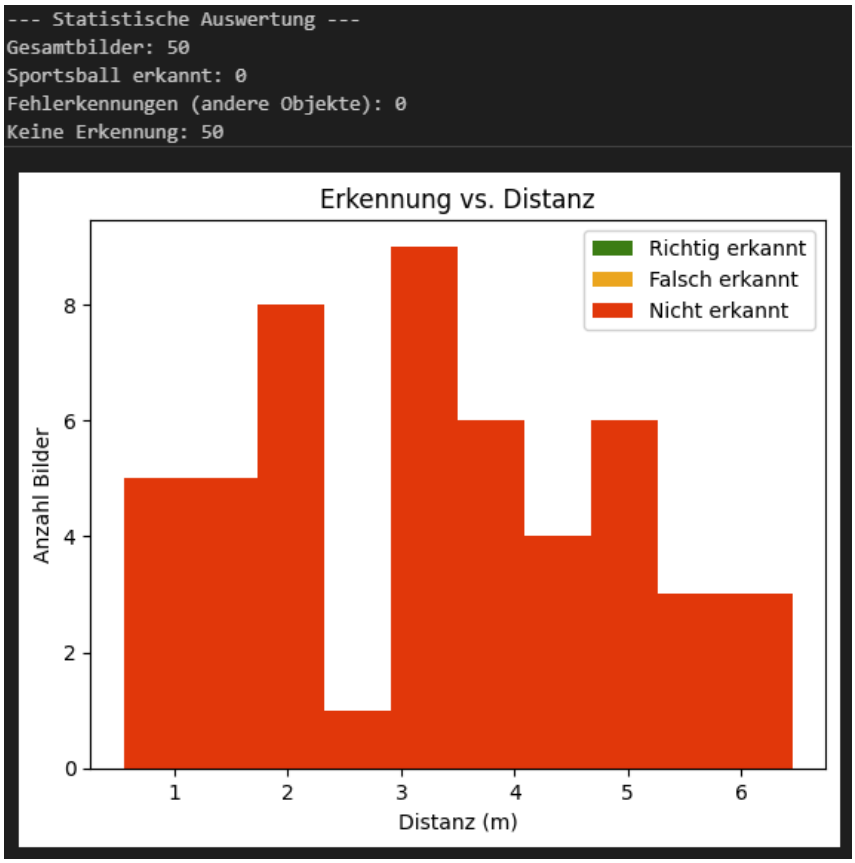

Figure 5: Statistische Auswertung der richtig, falsch und nicht erkannten Tennisbälle als Sportsball Objekt auf der weißen Linie.

Eine Vermutung war, dass die Platzierung des Tennisballs direkt auf der weißen Linie problematisch war, da sich der Ball nicht gut von dieser abheben konnte, siehe Figure 6. Desweiteren ist erkennbar, dass sich ein Schatten auf der unteren Hälfte des Balles abzeichnet, dieser könnte die Erkennung des Balls auch beeinträchtigen-

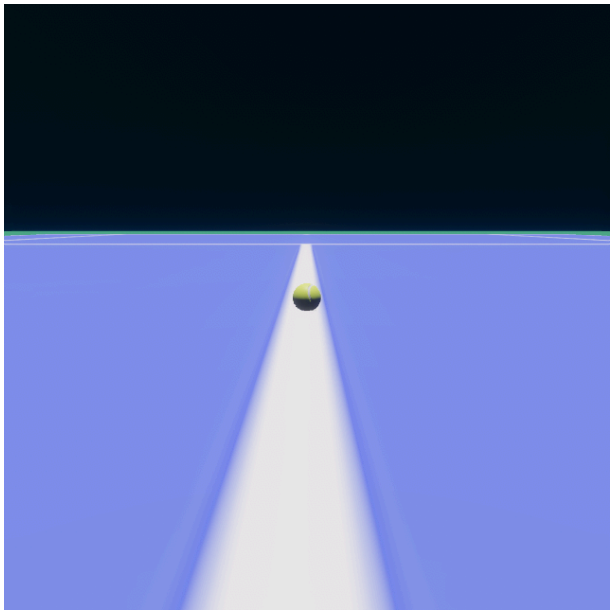

Figure 6: Beispielbild des Balls auf der weißen Linie.

Der Ball und die Kamera wurden weiter nach rechts bewegt, so, dass der Ball nun vor blauem Hintergrund lag, wie auf Figure 7 zu erkennen. Außerdem wurden die Materialien des balls mit “unlit” Farben ersetzt, sodass eine Schattenbildung auf dem Ball nicht möglich ist.

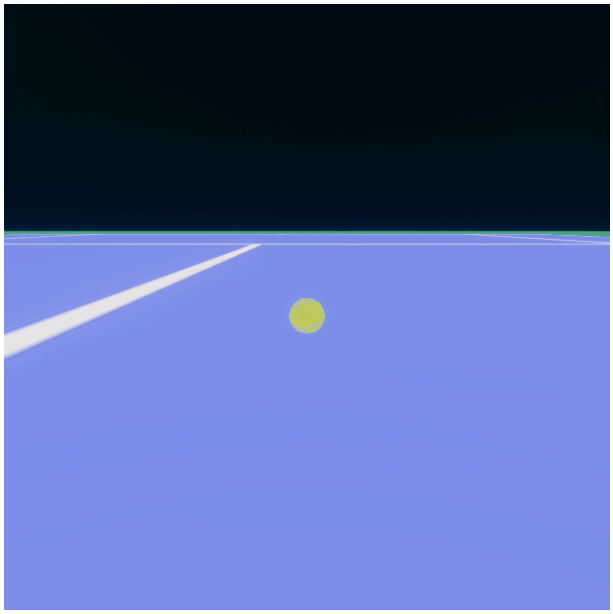

Figure 7: Beispielbild des Balls auf der weißen Linie.

Nun erkannte Yolo bereits 23 der 50 Tennisbälle, wie in Figure 8 zu sehen.

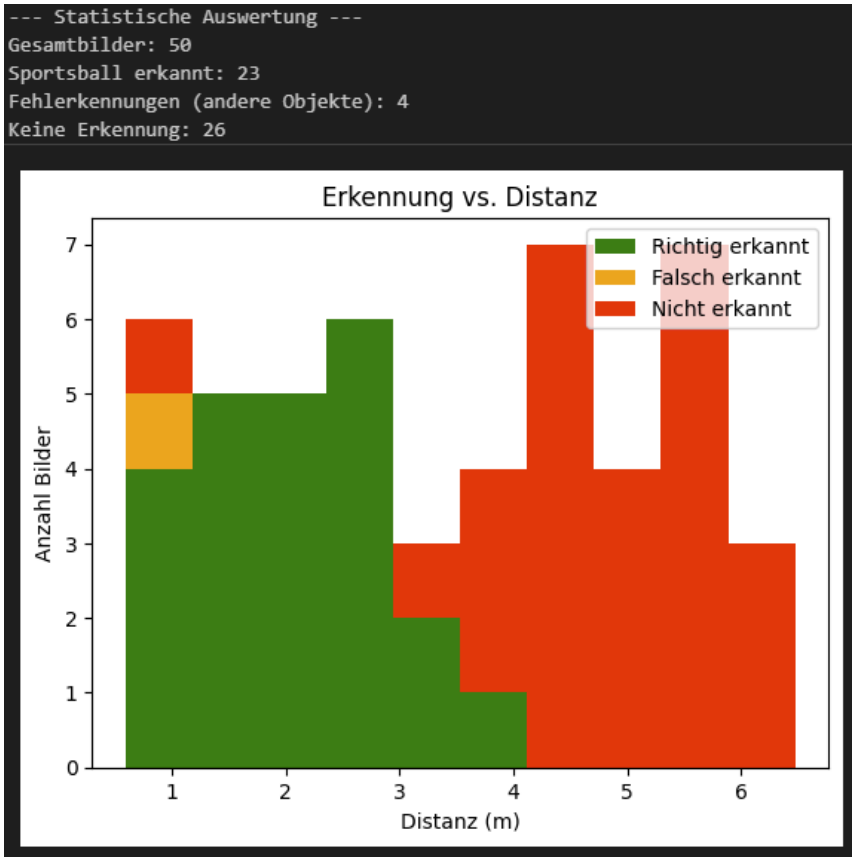

Figure 8: Statistische Auswertung der richtig, falsch und nicht erkannten Tennisbälle als Sportsball Objekt auf blauem Hintergrund.

Aus Figure 8 ist ersichtlich, dass die Entfernung des Balles eine wichtige Rolle bei der erfolgreichen Erkennung des Balls als Sportsball spielt. Genauer scheint ab einer Entfernung des Balles von mehr als drei Meter zur Kamera die Menge an richtig erkannten Bällen auffallend gering zu werden.

Das ist nachvollziehbar wenn man berücksichtigt, dass der Ball in der Auflösung 640x640 ab einer bestimmten Entfernung nicht mehr wirklich als Ball erkennbar ist, siehe Figure 9.

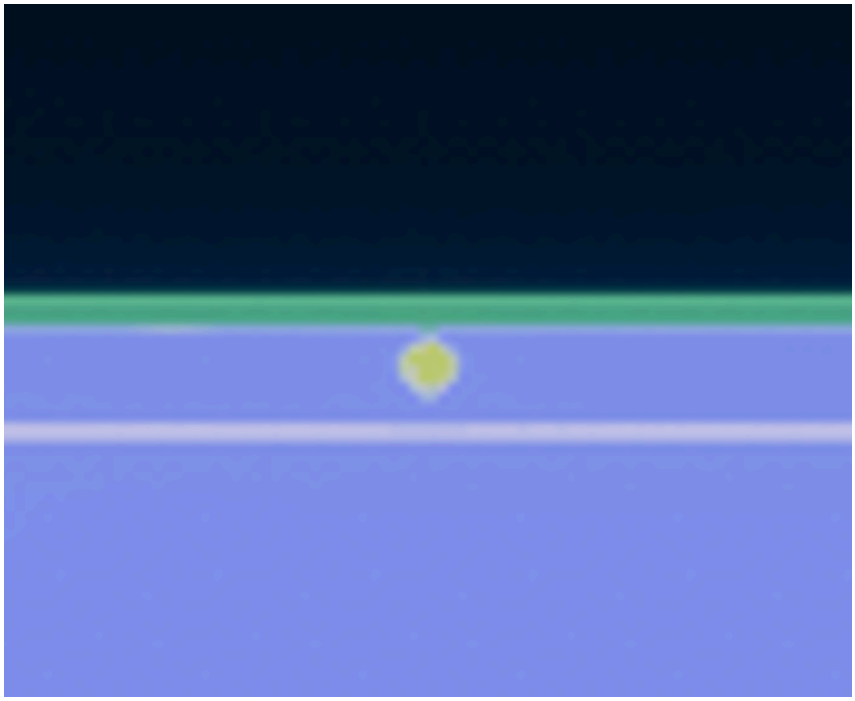

Figure 9: Hereingezoomter screenshot des Balles mit weiter Entfernung von ca. 5.83.

Auch bei einer höheren Anzahl an Screenshots, 200 in diesem Fall, bleibt es dabei, dass Distanzen über drei Meter dazu führen, dass der Ball nicht mehr richtig erkannt wird, siehe Figure 10.

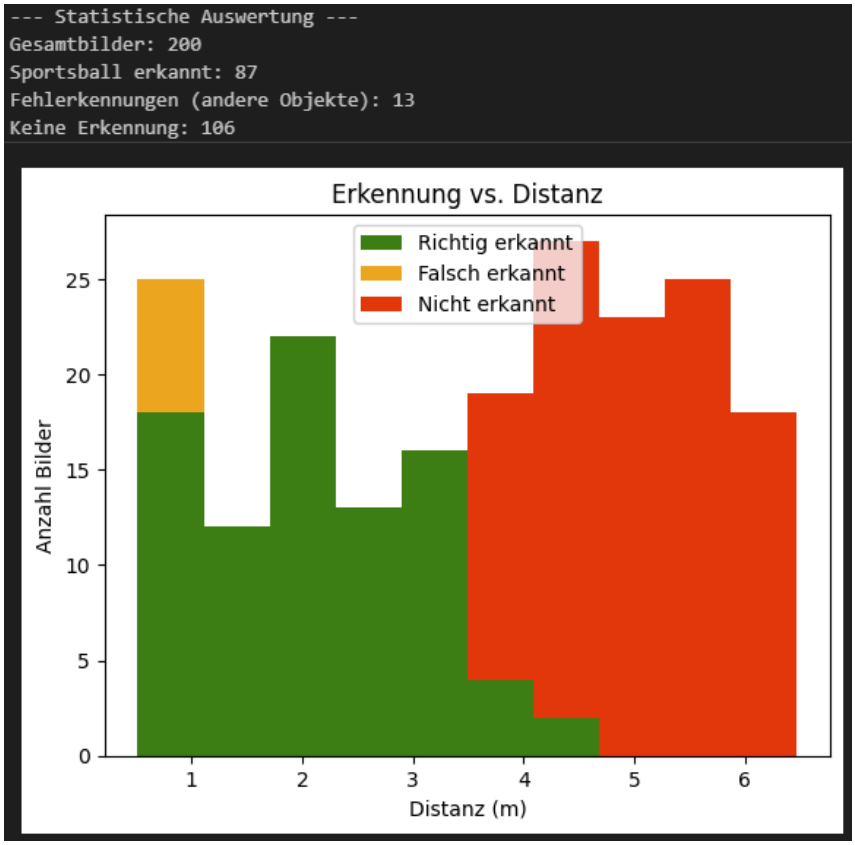

Figure 10: Statistische Auswertung der richtig, falsch und nicht erkannten Tennisbälle als Sportsball Objekt auf blauem Hintergrund mit 200 Screenhots.

Hier stellt sich nun die Frage, ob es überhaupt sinnvoll ist, eine so geringe Auflösung für ein Bild zu nutzen, wenn diese verhindert Bälle auf größeren Entfernungen erkennen zu können?

Jedoch ist anzumerken, dass es sich bei diesem Versuch nicht darum handelt, bereits die Entfernung des Balles zu schätzen. Die Erkennung des Balles ist aber sehr wohl ein wichtiger Bestandteil dieser Problemstellung und wenn Bälle nur bis zu einer bestimmten Entfernung erkannt werden können, stellt das ein Problem dar, da der Roboter im echten leben im Obtimalfall auch Bälle auf der gegenüberliegenden Seite des Platzes erkennen und ansteuern können soll.

Kurzfristig ergeben sich nun zwei Lösungsmöglichkeiten:

1. Die Auflösung der Bilder anheben, z.B. auf 1208 x 1208. Das würde die Rechenzeit anheben aber Bälle auf weiter entfernten Distanzen eventuell besser sichtbar machen.
2. Die Platzierung der Bälle auf ein Maximum von drei Metern beschränken.

Da die Frage dieser Arbeit ist, ob es möglich ist Distanzen mittels eines NN aus einem zweidimensionalen Bild heraus zu schätzen, werde ich mich für die zweite Möglichkeit entscheiden. Dafür werde ich das Script so anpassen, dass eine maximale Entfernung der Ballplatzierung von der Kamera als Parameter eingegeben werden kann.

Nach der Anpassung lässt sich eine wesentliche Verbesserung der Performance beobachten, siehe Figure 11.

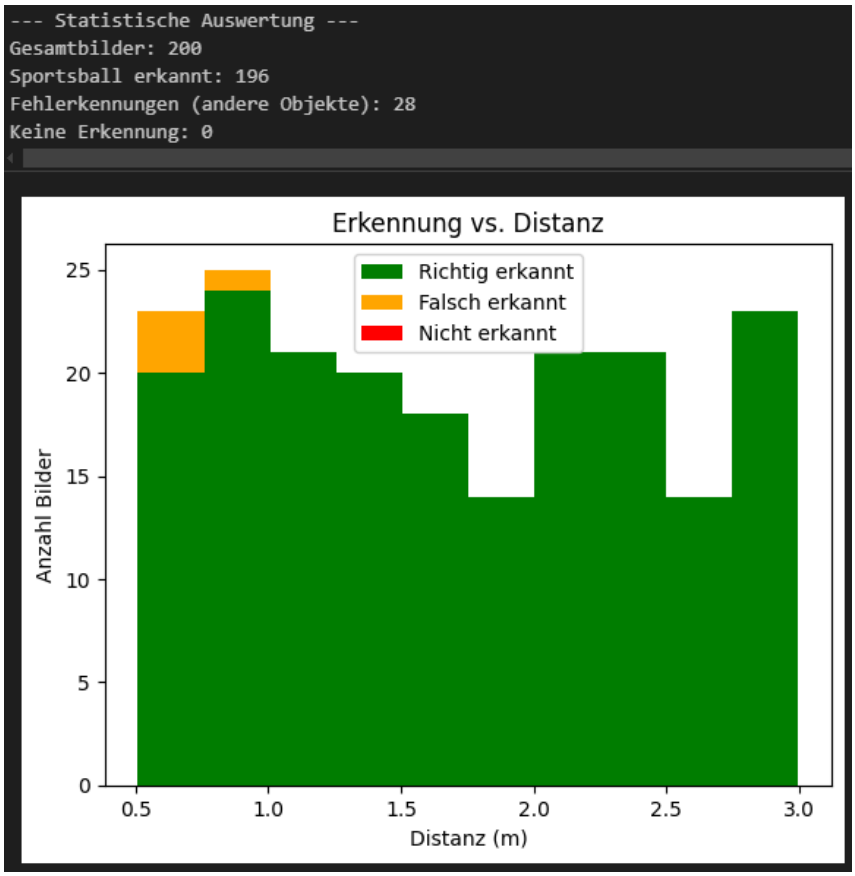

Figure 11: Statistische Auswertung der richtig, falsch und nicht erkannten Tennisbälle als Sportsball Objekt auf blauem Hintergrund mit 200 Screenhots und einer maximalen Entfernung des Balls von 3 Metern.

Als nächstes geht es daran, einen Datensatz zu schaffen mit genug Bildern für das Training eines NN. Zufällig wird hier die Zahl 2000 gewählt.

Jetzt müssen hier weitere Daten heraus gezogen werden, nämlich die Boundignbox koordinaten sowie dessen Größe, die jedoch auch aus den Koordinaten berechnet werden kann.

6. Feature Engineering

7. Modelltraining

Muss hier nur yolo auf die Bälle gefinetuned werden und dann simple Mathe als Algo? Oder kann/muss ich da auch ein Regressionsmodell bauen?

8. Modellevaluierung

9. Fazit

10. Bibliographie

[1] "Abmessungen Tennisplatz." Accessed: Jun. 11, 2025. [Online]. Available: https://www.tennisplanet.de/tennisberatung/abmessungen-tennisplatz

[2] Valentino, "Durchmesser vom normalen Tennisball | tennischeck.de." Accessed: Jun. 11, 2025. [Online]. Available: https://tennischeck.de/tennisball-durchmesser/

[3] "Camera - Raspberry Pi Documentation." Accessed: Jun. 11, 2025. [Online]. Available: https://www.raspberrypi.com/documentation/accessories/camera.html