Lehrstuhl VI für Künstliche Intelligenz und Angewandte Informatik

Dokumentation Masterpraktikum

Kicker-Balltracking

Maximilian Schmitt
20.3.2017

Inhalt

Abbildungen

1 Einleitung

Tischfußball ist eine Sportart, die auf einem Spielgerät namens Kicker oder Kickertisch gespielt wird. Das Spiel ahmt die bekannte Sportart Fußball in Miniaturform nach und hat sich seit seiner Erfindung Anfang des 20. Jahrhunderts zu einer der beliebtesten Kneipen-Sportarten entwickelt.

Auch am Lehrstuhl für Informatik VI an der Julius-Maximilians-Universität in Würzburg wird sehr gerne Tischfußball gespielt. Das Spiel wird hier allerdings hauptsächlich zu wissenschaftlichen Zwecken betrieben. Durch das Filmen des Spielfelds beispielsweise können interessante Daten gewonnen werden, die u.a. für die Objekterkennung in Bildern verwendet werden können. Aus dem Forschungsengagement des Lehrstuhls im Bereich der Objekterkennung in Bildern und der Leidenschaft am Tischfußball entstand dieses Masterpraktikum

In diesem Masterpraktikum geht es um die automatische Erkennung des Kicker-Balls auf Bildern von einem Kicker-Spielfeld während eines Spiels. Genauer geht es um die Implementierung einer Software, die auf gegebenen Bildern eines Kicker-Spielfelds möglichst weit automatisiert den Ball erkennt und darüber hinaus eine grafische Oberfläche bietet, mit der die Ergebnisse der automatischen Erkennung mit den tatsächlichen Positionen verglichen werden können. Die grafische Oberfläche soll den Benutzer möglichst gut beim manuellen Festlegen der Ballposition auf einer Menge von Bildern unterstützen und außerdem statistische Kennzahlen zum Auswerten verschiedener Daten- und Parametersätze veranschaulichen.

2 Verwandte Arbeiten

Es gibt bereits einige wenige Arbeiten, die sich mit dieser bzw. einer sehr ähnlichen Thematik befassen. In der Arbeit von Janssen, de Best und van de Molengraft mit dem Titel „Real-Time Ball Tracking in a Semi-automated Foosball Table[1]“ geht es beispielsweise um die Erkennung des Kicker-Balls in Echtzeit. Hier wurde eigens ein halbautomatischer Kickertisch konstruiert, der ein fest verbautes Gestell zum Halten der Kamera hat. Der Tisch wurde dafür konstruiert, dass ein menschlicher Spieler gegen einen Computergegner antritt. Die Erkennung der Position des Kicker-Balls auf dem Spielfeld dient dem Computer dabei als notwendige Eingabe, um die Bewegung der Stangen steuern zu können. Die Objekterkennung muss hierbei natürlich extrem schnell geschehen, da sich der Ball in der Regel sehr schnell über das Spielfeld bewegt und nach der Objekterkennung noch die Bewegung der Stangen gesteuert werden muss. Janssen et al. verwenden in ihrer Arbeit eine Vorgehensweise, bei der versucht wird aus den Bildern die Objekte auf einem Kickertisch durch statische Masken herauszufiltern.

In der Arbeit von Bambach und Lee mit dem Titel „Real-Time Foosball Game State Tracking[2]“ geht es auch unter anderem um die Ermittlung der Position des Kicker-Balls auf dem Spielfeld in Echtzeit. Auch in dieser Arbeit wurde ein spezieller Tisch verwendet mit einem Aufbau für die Halterung der Kamera. Allerdings lag der Fokus hier auf der ganzheitlichen Ermittlung des Spiel-Status. Das schließt neben der

[1] Rob Janssen, Jeroen de Best, René van de Molengraft: „Real-Time Ball Tracking in a Semi-automated Foosball Table“. Springer-Verlag. 2010.
[2] Sven Bambach, Stefan Lee: „Real-Time Foosball Game State Tracking“. 2012.

Ballposition auch die Position und Rotation der Spieler auf den Stangen ein. Für die Erkennung des Balls verwendeten Bambach und Lee Hintergrund-Masken und Kalman-Filter.

3 Datengrundlage

Die Datengrundlage für das Praktikum besteht aus Videoaufzeichnungen von Kickerspielen am Lehrstuhl. Für diese Videos wurde der Kickertisch jeweils von oben gefilmt, sodass das Spielfeld immer komplett und möglichst immer an der gleichen Position innerhalb des Bildes zu sehen ist. Abbildung 1 zeigt ein Einzelbild aus einem der gegebenen Videos. Alle Videos wurden aus dieser Perspektive gefilmt.

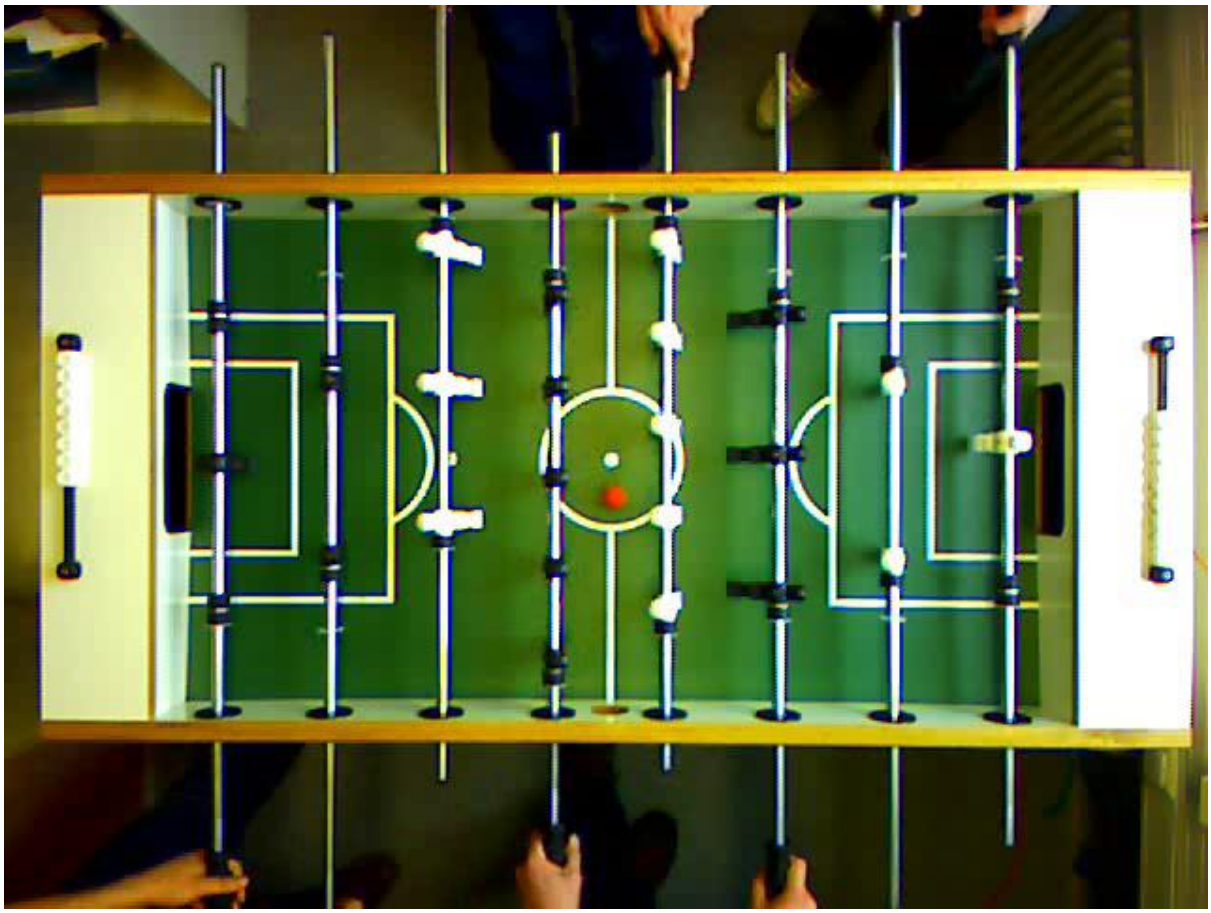

Abbildung 1: Beispiel eines Einzelbildes aus einem gegebenen Video

Aus den gegebenen Videos wurden anschließend Einzelbilder extrahiert. Dies wurde mit dem Programm *FFmpeg*[3] erledigt. FFmpeg ist eine freie Software zum Konvertieren von Multimedia-Dateien in verschiedenste Formate. Unter anderem bietet das Tool die Extraktion von Einzelbildern aus Video-Dateien.

[3] Details zu FFmpeg sind unter https://ffmpeg.org/ zu finden.

In diesem Praktikum wurden die Bilder mit einer Framerate von zehn Bildern pro Sekunde extrahiert. Bei verschiedenen Tests hat sich herausgestellt, dass sich mit Erhöhung der Rate keine besseren Ergebnisse beim automatischen Erkennen der Ballpositionen erzielen lassen. Wählt man hingegen eine kleinere Rate, springt die Ballposition sehr oft hin und her, sodass der implementierte Algorithmus größere Probleme beim Markieren des Balls hatte.

4 Implementierung

Die Implementierung kann grob in zwei Hauptbestandteile aufgeteilt werden. Zum einen wurde eine grafische Oberfläche implementiert, die den Nutzer von der Auswahl der Bilder über das händische und automatische Markieren des Balls bis hin zur statistischen Auswertung unterstützt. Zum anderen wurde ein eigens entwickelter Algorithmus implementiert, der automatisiert versucht die richtige Ballposition innerhalb eines Bildes zu ermitteln.

Im Folgenden wird nun im Detail auf beide Implementierungen eingegangen.

4.1 Grafische Oberfläche

Die implementierte Benutzeroberfläche dient folgenden Zwecken:

- Veranschaulichung der Ergebnisse des implementierten Algorithmus zur automatischen Ballerkennung
- Einfache und schnelle Navigation zwischen Datensätzen und Bildern innerhalb eines Datensatzes
- Unterstützung des Benutzers beim manuellen Setzen der Ballposition innerhalb der Bilder
- Import- und Export-Funktion für bereits gesetzte Ballpositionen
- Veranschaulichung statistischer Kennzahlen für den Vergleich zwischen manueller und automatischer Markierung der Ballpositionen

4.1.1 Allgemeines

Die Benutzeroberfläche lädt bei Programmstart ein Verzeichnis, in dem alle Bilder des zu bearbeitenden Datensatzes liegen. Die Bilder sollten dabei in alphabetischer Reihenfolge so benannt sein, dass die Reihenfolge dem Videoverlauf entspricht. Das Verzeichnis ist in den Programmparametern vor Programmstart festzulegen (siehe Kapitel 4.3).

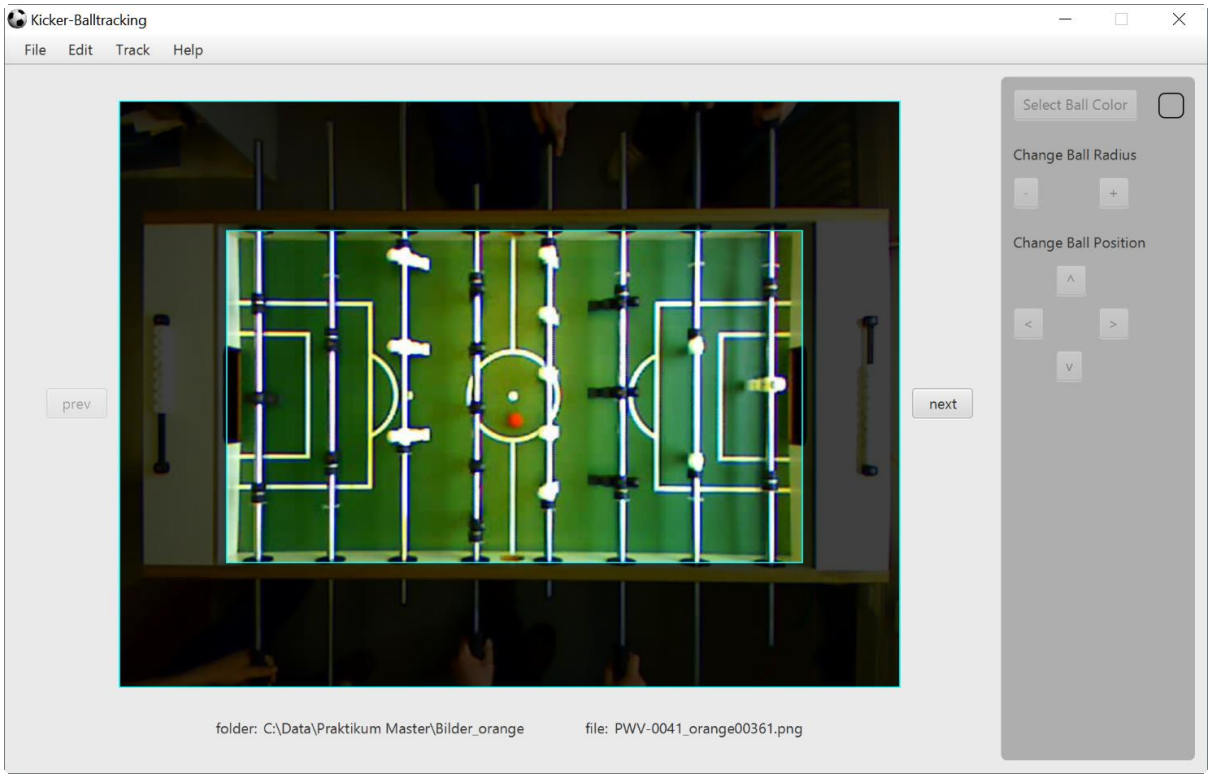

Abbildung 2: Benutzeroberfläche nach Programmstart

Abbildung 2 zeigt die Benutzeroberfläche nach Programmstart. Mittig ist das erste Bild im vorausgewählten Verzeichnis zu sehen. Die beiden Beschriftungen „folder“ und „file“ kennzeichnen das aktuell ausgewählte Bildverzeichnis bzw. den Dateinamen des aktuell zu sehenden Bildes. Mit den Buttons „prev“ und „next“ kann das Bild in der Reihenfolge vor bzw. nach dem aktuellen Bild angezeigt werden. Alternativ zu den Buttons „prev“ und „next“ kann auch mit den Tasten F3 (zurück) und F4 (vorwärts) durch die Bilder im ausgewählten Verzeichnis navigiert werden. Der blaue Rahmen, der in der Abbildung auf dem Bild zu sehen ist und den Bereich außerhalb des Spielfelds abdunkelt, dient dazu das Spielfeld – und damit den Suchraum für die Ballposition – noch genauer abgrenzen zu können. Die genaue Bedeutung des Rahmens wird in Kapitel 4.2 erläutert.

Verzeichnis- und Bildwechsel

Über das Menü „File“ kann sowohl das Bildverzeichnis (Menüpunkt „Open Folder“) als auch das aktuell angezeigte Bild (Menüpunkt „Open Image“) zur Laufzeit gewechselt werden. Der Verzeichniswechsel dient hauptsächlich zum Wechsel zwischen verschiedenen Datensätzen, während die Auswahl des aktuellen Bildes den Benutzer gerade bei sehr langen Videos mit vielen Einzelbildern unterstützt, um auch innerhalb eines Datensatzes bestimmte Szenen konkret ansteuern zu können.

4.1.2 Tracking-Workflow

Initialisierung des Tracking-Laufs

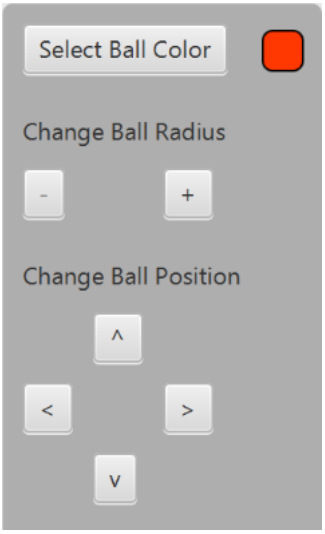

Abbildung 3: Schaltflächen zur genauen Festlegung der initialen Ballposition, der Größe und Farbe des Balls

Zu Beginn eines Tracking-Laufs muss der Benutzer über den Menüpunkt „Track“ → „Init Tracking“ für den Algorithmus zur automatischen Ballerkennung die Position des Balls auf dem aktuellen Bild festlegen. Hierzu muss zunächst per Linksklick die Ballposition ermittelt (nach Auswahl des Menüpunkts „Init Tracking“ hat der Maus-Cursor bis zur Festlegung der Position auf dem Bild die Form eines Fadenkreuzes). Hierbei wird nicht nur die Position des Balls, sondern auch dessen Farbe und Größe (Radius) festgelegt bzw. ermittelt. Die Farbe und Größe des Balls wird als Eingabe für den Algorithmus zur automatischen Erkennung des Balls benötigt. Über die Buttons, die in Abbildung 3 zu sehen sind, können Farbe, Größe und Position des Balls noch genau justiert werden. Die Position, die markiert wurde, wird stets mit einem Kreis in Größe des Balls gekennzeichnet, um dem Benutzer zu zeigen, welche Position aktuell ausgewählt ist.

Nachdem Form und Position des Balls auf dem ersten Bild festgelegt wurden, muss die Initialisierung über den Menüpunkt „Track“ → „Finish Tracking“ abgeschlossen werden. Mit Abschluss der Initialisierung sind Farbe und Größe des Balls für den weiteren Verlauf des Tracking-Laufs unwiderruflich festgelegt.

Manuelles Tracking

Mit der implementierten Benutzeroberfläche hat der Benutzer die Möglichkeit, für jedes Bild im ausgewählten Verzeichnis die Position des Balls manuell festzulegen, um einen Vergleichswert für die automatische Ballerkennung zu liefern. Hierzu muss der Menüpunkt „Track“ → „Track Manual“ ausgewählt werden oder wahlweise die Taste F5 gedrückt werden. Anschließend erscheint der Maus-Cursor wie schon bei der Initialisierung der Ballform als Fadenkreuz über dem Bild bis die Position per Linksklick festgelegt wurde. Nach dem ersten Festlegen der Position erscheint wieder eine Markierung (standardmäßig in blau) in der Größe des Balls. Falls die Position nicht gewünscht durch den Mausklick getroffen wurde, kann auch hier jederzeit die Position über die Pfeil-Buttons am rechten Rand des Fensters genau angepasst werden. Alternativ zu den angesprochenen Pfeil-Buttons kann auch hier per Tastatur-Abkürzung die Position angepasst werden, um effizienter arbeiten zu können. Hierzu muss STRG in Verbindung mit den Pfeiltasten verwendet werden. Falls der Ball auf einem Bild überhaupt nicht zu sehen ist (z.B. weil gerade ein Tor erzielt wurde), kann mit Klick auf den abgedunkelten Bereich ein spezieller Positionswert gesetzt werden, der angibt, dass der Ball nicht auf dem Bild zu sehen ist.

Automatisches Tracking

Eine der Hauptfunktionen ist die automatisierte Erkennung der Ballposition innerhalb der Bilder. Dieser Schritt kann auf zwei Arten in der Benutzeroberfläche gesteuert werden:

1. Halbautomatisch:
   Hierbei muss für jedes Bild der Menüpunkt „Track“ → „Track Next“ (wahlweise auch über den Shortcut F6 ausführbar) ausgewählt werden. Es wird dann auf dem nächsten noch nicht automatisch bearbeiteten Bild versucht mit dem implementierten Algorithmus (siehe Kapitel

4.2) die Ballposition zu ermitteln. Falls eine Ballposition für ein Bild gefunden werden konnte, wird eine kreisförmige Markierung mit Ballradius angezeigt (standardmäßig grün).

2. Vollautomatisch:
   Hierbei muss lediglich einmal der Menüpunkt „Track“ → „Track All“ (wahlweise auch F8) ausgewählt werden. Es wird dann für alle Bilder in der Reihenfolge nach dem ersten Bild (Bild, das für die Initialisierung verwendet wurde) automatisch versucht die richtige Ballposition zu ermitteln. Hierzu erscheint ein Fortschritts-Dialog mit wessen Hilfe abgeschätzt werden kann, wie lange der Prozess noch dauert (siehe Abbildung 4). Außerdem kann das automatische Tracking über den Button „cancel“ abgebrochen werden.

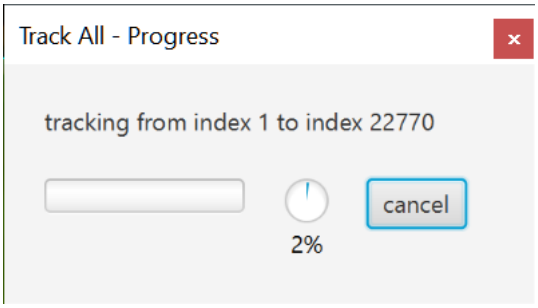

Abbildung 4: Fortschritts-Dialog für den Menüpunkt "Track All"

4.1.3 Veranschaulichung, Auswertung und sonstige Funktionen

Anzeige des Positionsverlaufs

Über den Menüpunkt „Edit“ → „Show Path“ kann man sich zusätzlich zu den Markierungen für das manuelle und automatische Tracking den Positionsverlauf der bisher ermittelten Positionen auf dem aktuell ausgewählten Bild anzeigen lassen. Abbildung 5 zeigt den Positionsverlauf für eine Menge automatisch ermittelter Ballpositionen. Die grünen Kreise stellen die einzelnen Positionen dar. Die rote Linie dient dazu, besser einen Zusammenhang und einen zeitlichen Verlauf erkennen zu können.

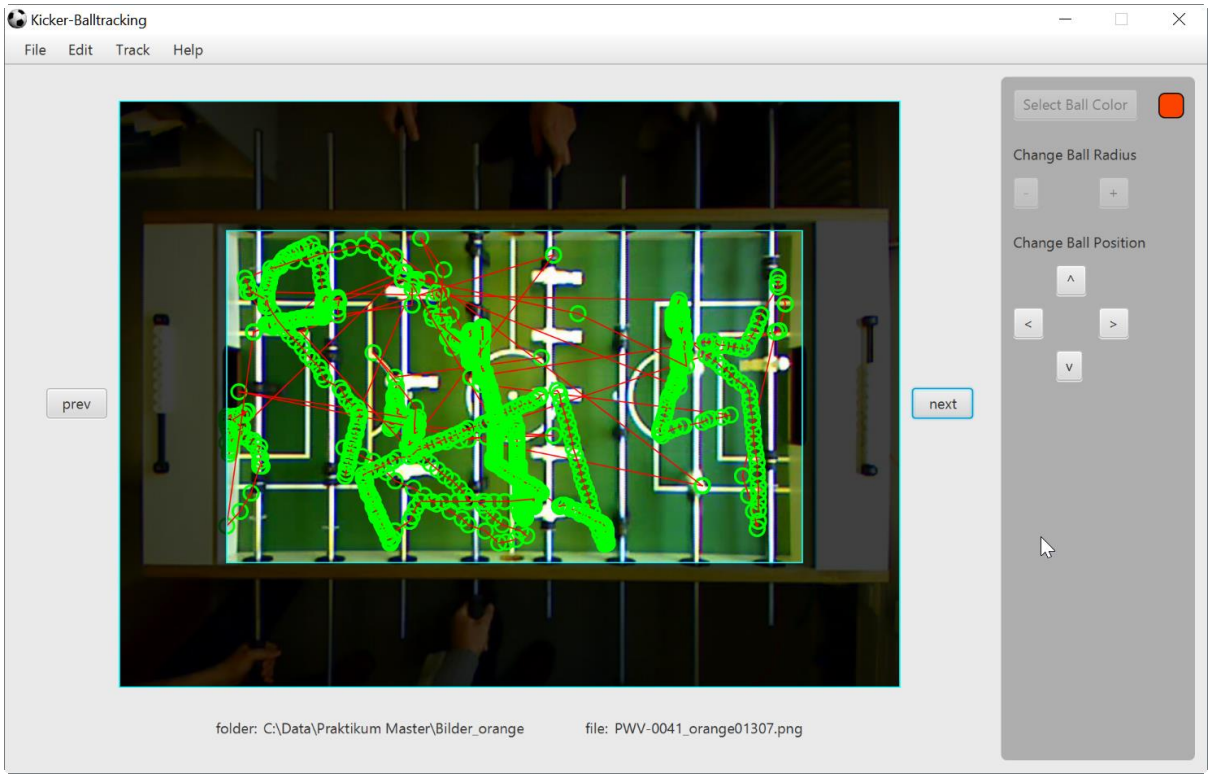

Abbildung 5: Anzeige des Positionsverlaufs automatisch ermittelter Ballpositionen

Statistische Auswertung

Ein wichtiger Bestandteil der Benutzeroberfläche ist die statistische Auswertung der Ergebnisse eines Tracking-Laufs. Hier kann ein Lauf bestehend aus manuell und automatisch ermittelten Ballpositionen anhand diverser Kennzahlen analysiert werden. Die statistische Auswertung wird nach der manuellen und automatischen Ermittlung der Ballpositionen über den Menüpunkt „Edit“ → „Show Statistics“ in einem separaten Dialog angezeigt.

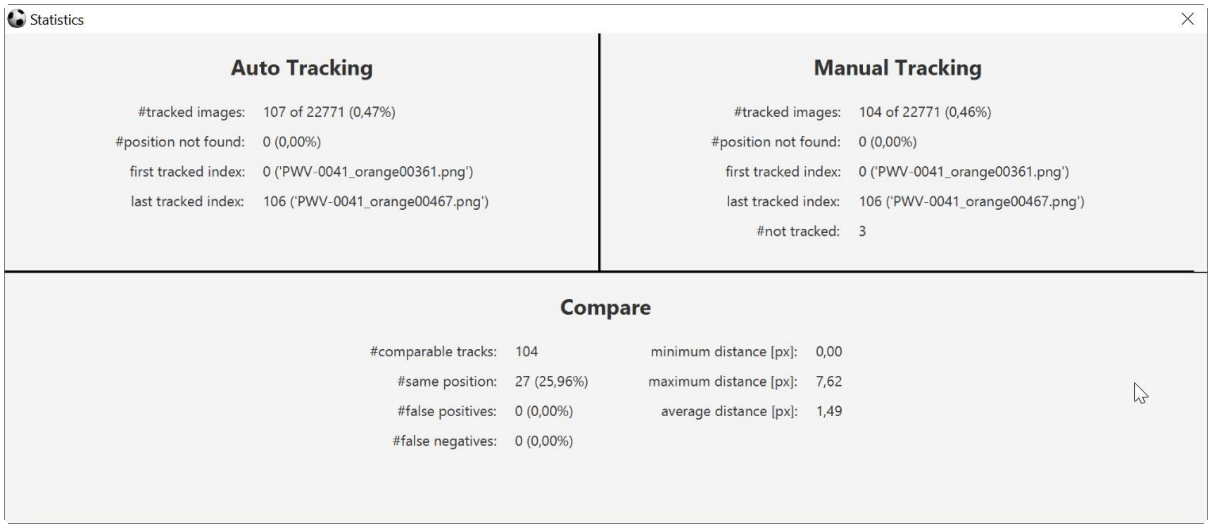

Abbildung 6: Statistische Auswertung eines Tracking-Laufs

In Abbildung 6 ist ein Beispiel einer statistischen Auswertung zu sehen. Im oberen Teil werden Kennzahlen für die automatische Ballerkennung (links) und die manuelle (rechts) dargestellt. Folgende Kennzahlen sind vorhanden:

- #tracked images: Anzahl der Bilder, für die eine Ballposition ermittelt wurde (Prozentzahl bezieht sich auf den gesamte Datensatz, d.h. alle Bilder im ausgewählten Verzeichnis)
- #position not found: Anzahl der Bilder, für die keine Ballposition ermittelt werden konnte
- #not tracked: Anzahl der Bilder zwischen dem ersten und dem letzten bearbeiteten Bild, für die beim manuellen Positionserfassen keine Position angegeben wurde

Im unteren Bereich der Statistik sieht man den Vergleich zwischen manuell und automatisch erfassten Ballpositionen mit folgenden Kennzahlen:

- #comparable tracks: Anzahl der Bilder, für die sowohl manuell als auch automatisch eine Ballposition ermittelt wurde
- #same position: Anzahl der Bilder, bei denen die Positionen identisch waren (Prozentzahl in Bezug auf #comparable tracks)
- #false positives: Anzahl der Bilder, für die automatisch eine gültige Position ermittelt wurde, allerdings manuell die spezielle Position „nicht gefunden“ ermittelt wurde
- #false negatives: Anzahl der Bilder, für die manuell eine gültige Position ermittelt wurde, allerdings automatisch die spezielle Position „nicht gefunden“ ermittelt wurde
- minimum distance [px]: minimale Distanz zwischen manuell und automatisch ermittelter Position auf einem Bild in Pixel
- maximum distance [px]: maximale Distanz zwischen manuell und automatisch ermittelter Position auf einem Bild in Pixel
- average distance [px]: durchschnittliche Distanz zwischen manuell und automatisch ermittelter Position auf einem Bild in Pixel

Persistieren eines Tracking-Laufs

Sowohl die Ballpositionen, die manuell ermittelt wurden, als auch die automatisch gefundenen Ballpositionen werden über komplette Dauer einer Sitzung im Hauptspeicher gehalten und können über die mitgelieferte Export-Funktion persistiert werden. Diese Funktion wird über den Menüpunkt „File“ → „Export“ angesteuert. Anschließend muss man lediglich ein Ziel-Verzeichnis und einen Dateinamen für die XML-Datei, die von der Software generiert wird, auswählen und mit „speichern“ bestätigen.

Import eines persistierten Tracking-Laufs

Umgekehrt kann auch ein generiertes XML eines vergangenen Tracking-Laufs wieder eingelesen und fortgesetzt bzw. ausgewertet werden. Hierzu wird über den Menüpunkt „File“ → „Import“ die Import-Funktion aufgerufen. Es muss dabei lediglich ein XML im entsprechenden Format im Dateisystem ausgewählt werden und mit „öffnen“ bestätigt werden.

Hilfe

Um dem Benutzer mit einfachen Mitteln eine Dokumentation bereitzustellen, wurde ein weiteres Menü „Help“ in die Menüleiste eingefügt. Klickt man auf den Menüeintrag „Help“ → „Open doku“ wird mit dem assoziierten Standardprogramm für PDF-Dateien eine Kopie dieser Dokumentation geöffnet.

Tipps zur Bedienung

Grundsätzlich ist es empfehlenswert die angebotenen Tastatur-Shortcuts zu verwenden, da dadurch viel Zeit bei der Bedienung der Benutzeroberfläche gespart werden kann.

Will man bei einem Datensatz verschiedene Parameter für die automatische Ermittlung der Ballpositionen testen und diese jeweils mit manuell erfassten Positionen vergleichen, lohnt sich die Benutzung der Export- und Import-Funktion. Man kann beispielsweise den kompletten Datensatz einmal nur manuell durcharbeiten und die Ergebnisse im Anschluss exportieren. Danach kann dieser Export für die vor der Durchführung der automatischen Läufe jeweils importiert werden. Dadurch muss man lediglich einmal pro Datensatz die Ballpositionen manuell erfassen.

4.2 Tracking-Algorithmus

Der implementierte Algorithmus zum automatischen Erkennen der Ballposition in Bilder eines Kickertischs basiert hauptsächlich auf der Berechnung eines „Negativbildes“ anhand eines gegebenen Farbschwellwertes, der den maximalen Farbabstand zur Ballfarbe angibt. Außerdem geht der Algorithmus von der Annahme aus, dass die Wahrscheinlichkeit sehr hoch ist, dass sich die Position des Balls zur Position des Balls auf dem vorherigen Bild nur sehr geringfügig ändert. Als zwingende Voraussetzung wird für den Algorithmus die Angabe der Farbe des Kicker-Balls und die Größe (Radius des Balls) benötigt.

Farbdistanz

Die Farbdistanz wird hier als euklidische Distanz zwischen zwei Farbvektoren mit jeweils drei Skalaren (rot, grün und blau) aufgefasst. Jeder Skalar hat dabei einen Wertebereich von 0 bis 255 (2^8 Werte).

Negativbild

Das Negativbild wird wie folgt berechnet:

1. Setze jedes Pixel innerhalb des Spielfelds (das Spielfeld ist durch blaue Box festgelegt) genau dann auf weiß, wenn die Farbdistanz zwischen der originalen Pixelfarbe und der Ballfarbe nicht größer als die gegebene maximale Distanz ist
2. Setze alle anderen Pixel auf schwarz

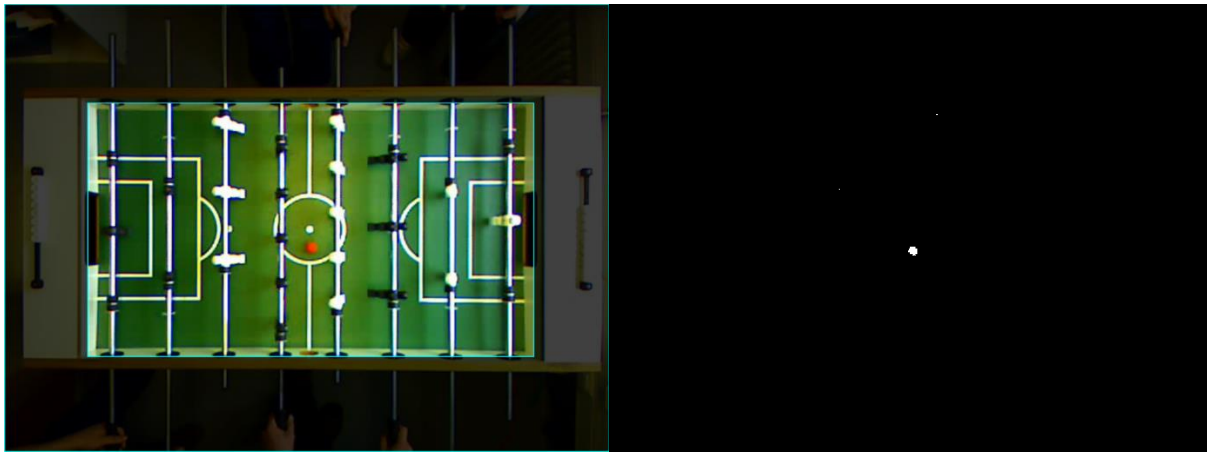

Abbildung 7: Beispiel eines Originalbildes mit entsprechendem Negativbild

Abbildung 7 zeigt ein Bild mit dazugehörigem Negativbild. Auf dem Negativbild ist ein großer weißer Fleck zu sehen, da hier im Originalbild der Ball ist und nicht verdeckt wird. Außer dem großen weißen Fleck an der Position des Balls im Bild sind nur vereinzelt weiße Pixel zu sehen.

4.2.1 Grundgerüst des Algorithmus

Der Algorithmus hat folgendes grobes Grundgerüst:

1. Berechne das Negativbild anhand der gegebenen maximalen Farbdistanz zur Ballfarbe
2. Ermittle die letzte Ballposition, die nicht „nicht gefunden“ ist
3. Falls die letzten x Suchen (x ist durch einen Schwellwertparameter gegeben) fehlgeschlagen sind oder die Ballposition aus Schritt 2 „nicht gefunden“ ist, suche ausgehend von der Mitte des Spielfelds kreisförmig im Negativbild nach dem Ball (gekennzeichnet durch mehrere weiße Pixel auf einem Fleck) und verwende die gefundene Position im weiteren Verlauf als vorherige Position
4. Falls die vorherige Ballposition immer noch „nicht gefunden“ ist, gebe „nicht gefunden“ zurück
5. Falls die vorherige Ballposition auf einem weißen Fleck im Negativbild ist, gebe das Zentrum des weißen Flecks als Position zurück
6. Ansonsten suche kreisförmig mit einem kleinen Radius nach der neuen Ballposition im Negativbild (gekennzeichnet durch mehrere weiße Pixel auf einem Fleck)
7. Falls in Schritt 6 eine mögliche Ballposition (weißer Fleck im Negativbild) ermittelt werden konnte, gebe das Zentrum des weißen Flecks als Position zurück
8. Falls in Schritt 6 keine mögliche Ballposition ermittelt werden konnte, gebe „nicht gefunden“ zurück

4.2.2 Parameter

Der Algorithmus verwendet folgende Parameter:

- *searchFailThreshold*:
  Schwellwert für Schritt 3 im Grundgerüst. Wenn mindestens die letzten *searchFailThreshold* Positionen, die ermittelt wurden, „nicht gefunden“ sind… Für diesen Parameter hat sich der Wert 4 als besonders gut herausgestellt.
- *radiusSearchSmall:*
  Gibt den Suchradius für die lokale Suche im Negativbild in Schritt 6 an. Hier hat sich der Wert 50 als besonders gut herausgestellt.
- *maxColorDistance:*
  Schwellwert für die Farbdistanz zwischen Ballfarbe und originaler Pixelfarbe bei der Erzeugung der Negativbilder. Der Wert 75 hat sich hier als besonders gut herausgestellt

Alle empfohlenen Parameter-Werte sind natürlich abhängig vom verwendeten Datensatz und können nicht in jeder Datenkonstellation das beste Ergebnis erzeugen.

4.2.3 Herausfordernde Situationen

Der implementierte Tracking-Algorithmus funktioniert bereits sehr gut, wenn der Ball nicht von den Figuren oder den Stangen verdeckt oder mit sehr hohem Tempo geschossen wird, sodass man den Ball mit bloßem Auge kaum noch erkennen kann.

Folgende Situationen stellen den Algorithmus allerdings bislang noch vor Probleme:

Ball nicht auf dem Spielfeld

Der einfachste Problemfall ist der Fall, wenn sich der Ball tatsächlich nicht auf dem Spielfeld befindet und somit keine Position gefunden werden sollte. Hierbei kommt es öfter vor, dass ein anderer kleiner Fleck, der eine ähnliche Farbe wie der Ball hat, fälschlicherweise als Ball erkannt wird.

Objekte mit ähnlicher Farbe auf dem Spielfeld

Grundsätzlich hat der Algorithmus die größten Probleme, wenn Objekte mit ähnlicher Farbe wie die Farbe des Balls auf dem Spielfeld zu sehen sind. Tests mit einem Video, bei dem der Ball weiß ist, haben sehr schlecht Ergebnisse geliefert, da die weißen Spielfeldmarkierungen und die silbernen Stangen einen sehr ähnlichen Farbton haben und nicht anhand der Farbe herausgefiltert werden konnten.

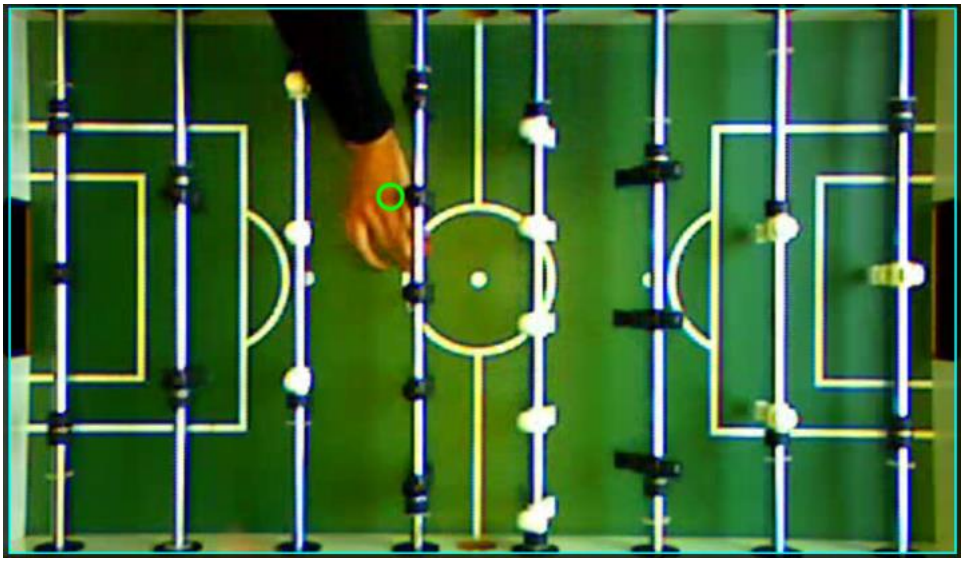

Abbildung 8: Beispiel einer fehlerhaften Erkennung. Die Hand eines Spielers wurde fälschlicherweise als Ball erkannt.

In Abbildung 8 ist ein Beispiel einer fehlerhaften Erkennung zu sehen. Hier wurde die Hand einer der Spieler als Ball erkannt, da der Ball die Farbe Orange hat und somit der Hand stark in der Farbe ähnelt.

Schnelle Schüsse

Außerdem stellen sehr schnelle Schüsse Probleme dar, da hier der Ball teilweise fast komplett verschwindet und nur noch ein langgezogener Schatten des Balls zu sehen ist. Dieser Schatten hat allerdings weder die Form noch die Farbe eines Balls und kann deshalb oft nur schwer vom Algorithmus erkannt werden.

4.3 Implementierungsdetails und Starten des Programms

Sämtliche Programmteile, die Teil dieses Masterpraktikums sind, wurden in JAVA (Compiler-Version 1.8) geschrieben. Die grafische Oberfläche wurde mit Hilfe des JAVA-eigenen GUI-Frameworks *JavaFX*[4] implementiert, das seit Version 8 fester Bestandteil des JDK ist. JavaFX bietet vor allem eine moderne Optik im Vergleich zum veralteten GUI-Framework Swing, das bisher oft verwendet wurde.

Außerdem wurde das Build-Management-Tool *Maven*[5] in der Version 3.3.9 eingesetzt, um die Abhängigkeiten zu weiteren Bibliotheken zu verwalten und um den Build-Prozess zu vereinfachen. Durch die Verwendung von Maven kann mit einem Befehl auf der Kommandozeile aus dem Source-Code ein fertiges Release mit allen benötigten Skripten und Dateien gebaut werden.

Verzeichnisstruktur des Builds

[4] Details zu JavaFX unter http://docs.oracle.com/javase/8/javase-clienttechnologies.htm .
[5] Details zu Maven unter https://maven.apache.org/ .

Ausgeliefert wird die komplette Verzeichnisstruktur des verwendeten GIT-Repositories. Relevant für den Anwender ist allerdings lediglich der Ordner „release“, da hier bereits ein fertiger Build liegt.

Im Ordner „release“ herrscht folgende Verzeichnisstruktur:

- release
    - doku
        - Masterpraktikum Kicker-Balltracking.pdf
    - lib (hier liegen die verwendeten Bibliotheken)
    - properties
        - settings.properties
    - kicker-ball-tracking-1.0.0.jar
    - startup.bat

Programm-Parameter

Vor dem Start der Anwendung sind ein paar Parameter einzustellen. Diese Parameter sind in der Datei „properties/settings.properties“ vom Benutzer an das eigene System anzupassen. Folgende Parameter können verwendet werden:

- debug_mode: Gibt an, ob für das Logging die Konsole verwendet wird statt einer Datei (true oder false)
- image_dir: Pfad zum Ordner, der bei Programmstart geladen werden soll
- image_width: Angabe der Breite der Bilder in Pixel
- image_height: Angabe der Höhe der Bilder in Pixel
- ball_radius: Default-Radius für die Initialisierung eines Tracking-Laufs
- manual_color: Farbcode für die Markierungen manuell erfasster Ballpositionen (default blau)
- auto_color: Farbcode für die Markierungen automatisch erfasster Ballpositionen (default grün)
- search_fail_threshold: Siehe Kapitel 4.2.2
- max_color_distance: Siehe Kapitel 4.2.2
- radius_search_small: Siehe Kapitel 4.2.2
- left_bound: Abstand des linken Rands des Spielfelds vom linken Bildrand in Pixel
- right_bound: Abstand des rechten Rands des Spielfelds vom linken Bildrand in Pixel
- top_bound: Abstand des oberen Rands des Spielfelds vom oberen Bildrand in Pixel
- bottom_bound: Abstand des unteren Rands des Spielfelds vom oberen Bildrand in Pixel

Starten des Programms

Zum Start des Programms ist lediglich das Skript „startip.bat“ auszuführen. Vorher ist allerdings sicherzustellen, dass eine JAVA-Installation mit Version 8 installiert ist und das bin-Verzeichnis der JAVA-Installation in der PATH-Systemvariable aufgenommen ist, sodass der Befehl „java –version“ jederzeit erfolgreich ausgeführt werden kann.

5 Fazit und Ausblick

Für dieses Masterpraktikum wurde ein Programm entwickelt, mit dem es schnell und einfach möglich ist, sowohl automatisiert als auch manuell die Position eines Kicker-Balls innerhalb von Bildern eines Kicker-Spielfelds zu erfassen. Außerdem wurde das Programm mit einer Statistik-Komponente zum Vergleich zwischen automatisch und manuell erfassten Ballpositionen abgerundet.

Diese Arbeit zeigt unter anderem auch die Probleme, denen sich ein Algorithmus wie der hierfür implementierte stellen muss. Sowohl die vielen Objekte auf einem Kicker-Spielfeld als auch die schnellen Bewegungen des Balls können Probleme bei der Objekterkennung darstellen.

Zukünftige Arbeiten werden wohl ehr in die Richtung der Arbeit von Janssen et al. gehen. Hier wurde das System zur Objekterkennung direkt in ein Komplettsystem eingebettet, das zudem noch fähig ist, eine Mannschaft des Kickertischs komplett selbst zu steuern und so menschlichen Gegnern eine Herausforderung zu bieten. Wir dürfen gespannt sein, wann es wohl den ersten Computer-Weltmeister im Tischfußball geben wird.