Team – TargetTracker - Dart

INFM2300 - Teamprojekt - Wintersemester 2020/2021

Autoren: Andreas Voigt, Ricardo Lissner, Rituraj Singh
andreas.voigt@hochschule-stralsund.de
ricardo.lissner@hochschule-stralsund.de
rituraj.singh@hochschule-stralsund.de
Matrikel-Nr. (Andreas Voigt): 19398
Matrikel-Nr. (Ricardo Lissner): 19134
Matrikel-Nr. (Rituraj Singh): 19539

Eingereicht am: 26.02.2021

Eingereicht bei: Prof. Dr. rer. Nat. Christian Bunse

Zusammenfassung

Dieses Dokument beschreibt das Vorgehen zu der Herausforderung eine Smartphone Applikation zu erstellen, welche in der Lage ist beim Spiel Darts die Würfe automatisiert zu überwachen, zu erkennen und zu zählen. Hierbei wird auf den theoretischen Lösungsansatz eingegangen, sowie den Stand der praktischen Umsetzung und deren Voraussetzungen. Dieses Dokument soll ebenfalls dazu dienen, dieses Projekt fortzuführen, weshalb an entsprechenden Stellen mehr auf die technischen Details eingegangen wird.

1 EINLEITUNG

Der Dart – Sport genießt in vielen Kulturen der Erde ein hohes Ansehen und erfreut sich gleichzeitig einer großen Anhängerschaft. Während die Profispieler neben den Schwierigkeiten des Werfens eines Darts, auch die Punktzahlen der geworfenen Darts sowie noch die zu werfenden Punkte im Kopf ausrechen können, stellt es viele Gelegenheitsspieler vor eine größere Herausforderung. Dementsprechend existieren diverse Werkzeuge, um wenigstens den Rechenaufwand für den Gelegenheitsspieler zu minimieren. Viele dieser Tools setzen entweder ein spezielles Spielbrett voraus (elektronische Dartscheibe mit speziellen Pfeilen) oder benötigen manuelle Eingaben des jeweiligen Nutzers. Damit auch Turnier gerechte Dartschreiben von einer automatischen Bewertung der Würfe profitieren können, bedarf es visueller Auswertemethoden. Mit der Motivation eine Lösung für das automatische Erkennen und Zählen von Dartpfeilen für handelsübliche Dartboards zu erstellen, wird im Folgenden ein möglicher Lösungsansatz auf Basis einer Smartphone Applikation beschrieben und vorgestellt.

1.1 AUFGABENSTELLUNG „TARGETSCANNER“

Tracken von Trefferpunkten in Echtzeit

Gruppengröße: 3-6

Im Rahmen des Projekts sollen Videoaufnahmen eines Ziels analysiert und Trefferpunkte berechnet werden. Zu beobachtende Ziele können dabei bspw. Dartscheiben oder Zieleauflagen von Sportschützen (Bogen, Armbrust, etc.) sein. Treffer sollen sicher erkannt werden und im Anschluss soll eine Punktebewertung nach den jeweiligen Regeln erfolgen. In der Endausbaustufe soll die Auswertung in Echtzeit erfolgen und bereits während eines Durchgangs Feedback geben. Das System soll in Form einer App nutzbar sein.

Die eben beschriebenen Anforderungen sollten im Rahmen dieses Modules für die Anwendung des „Dart“ – Sportes umgesetzt werden. Folgende Aufgabenpunkte sollten im Rahmen der Bearbeitung berücksichtigt werden.

Aufgaben:

1. Bildanalyse
2. Trefferauswertung
    - Im Bsp.-Bild sind alle Treffer im gelben Bereich.
3. Ausschluss bereits vorhandener Treffer
    - “Alte Treffer” werden nicht berücksichtigt
    - Doppeltreffer sind zu vernachlässigen
4. Live-Betrieb

Stichworte: Android, Bildanalyse

Dauer: 1 Semester

2 THEORETISCHE LÖSUNG

2.1 BILDANALYSE

Ein Großteil der theoretischen Lösungen basiert auf der strukturellen Unterteilung der Dartscheibe in Bereiche. Diese Unterteilung sollte möglichst in Echtzeit auf den laufend aufgenommenen Videostream bzw. auf deren Frames angewendet werden können. Jeder dieser Bereiche bzw. Zonen besitzt gleichzeitig einen Punktwert im Reglement des Dart Spiels.

Es ist davon auszugehen, dass eine Kamera nicht frontal vor die Dartscheibe platziert werden kann, da diese sonst die Spielerwürfe beeinflussen würde. Daher werden Bildaufnahmen außerhalb der Wurfzone stattfinden, was wiederum schräge Aufnahmen des Dartboards zur Folge hat. Da eine schematische Anzeige der Treffer geplant ist und für die Verbesserung der Treffererkennung ebenfalls eine perspektivische Entzerrung der aufgenommenen Bilder benötigt wird.

2.1.1 Strukturelle Aufteilung der Dartscheibe

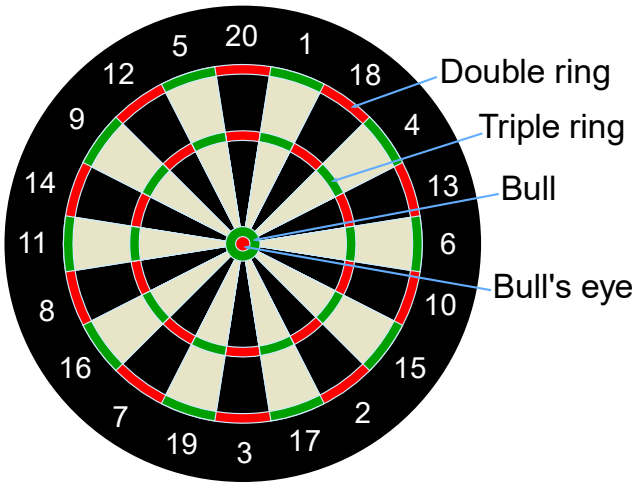

Abbildung 1 Schematische Darstellung einer Dartscheibe

Mit Hilfe von Bildanalyseverfahren wird die aufgenommene Dartscheibe in folgende Bereiche zerlegt:

- Bull's Eye
- Bull
- Double Ring
- Triple Ring
- Flächen
  - Grüne
  - Rote
  - Schwarze
  - Weiße
- Metalldrähte zwischen den Bereichen

Die Zerlegung findet hauptsächlich mit Hilfe von Farbmasken statt. Das heißt die Farbkanäle eines digitalen Bildes werden so manipuliert, dass nur noch die gesuchte Farbe als sichtbar erscheint. Das Ergebnis einer jeden Farbmaske wäre ein Schwarz-Weiß-Bild, wobei Weiß für die gesuchte Farbe steht und Schwarz für alle anderen Farben. Damit lassen sich Bitmasken erstellen, welche übereinandergelegt bzw. auf aufgenommene Bilder angewendet werden können. Für die Ringe bzw. Bull's Eye und Bull wird zusätzlich der Mittelpunkt des Bildes errechnet und mit Hilfe von Kontour- bzw. Ecken- und Kantenerkennung wird die Farbzerlegung vorgenommen.

2.1.2 Perspektivische Entzerrung

Die perspektivische Entzerrung bzw. das auch geometrische Transformation genannte Verfahren dient dazu, aus einer schrägen Aufnahme der Dartscheibe, diese wieder in eine Frontalansicht umzurechnen. Gesucht wird hierbei eine Transformationsmatrix, um von Bild A zu Bild B zu transformieren. Darüber hinaus können mit Hilfe eines maßstabsgetreuen Schemas einer Dartscheibe die einzelnen Bereiche (im Bild A) wieder festen Punktwerten zugeordnet werden (Bild B).

Für die Entzerrung gibt es unterschiedliche Ansätze, wobei im Wesentlichen zwei näher in Betracht kamen.

- Feature Matching
- Rektifizierung

Beim „Feature Matching" wird im Grunde versucht, ähnliche Bereiche auf 2 Bilder derselben Szene zu finden. Richtig angewendet bedarf dieses Verfahren keiner bzw. fast keiner menschlichen Hilfe. Hierfür gibt es verschiedene Algorithmen wie SIFT, SURF, FAST, ORB und weitere. Hierbei muss zunächst ein Ausgangsbild der Dartscheibe in Zentralperspektive aufgenommen und mit Koordinaten versehen werden. Danach dienen die Schrägaufnahmen als 2. Bild für's „Feature Matching."

Das Photogrammetrische Verfahren der Rektifizierung hat zum Ziel eine zentralperspektivische Projektion eines Bildes zu errechnen. Dieses Verfahren benötigt nicht zwangsläufig ein 2. Bild als Vergleich. Hierbei wären zwei Ansätze interessant:

- Interpolationsverfahren – Transformationen mit Hilfe von bekannten/festgelegten Passpunkten im Bild
- Parametrisches Verfahren – zu dem Aufgenommenen Bild sind Metainformationen bekannt wie Lage und Orientierung der Aufnahme im Raum

2.1.3 Pfeil- und Treffererkennung

Für die Erkennung der Dartpfeile soll das Verfahren der Semantischen Segmentierung erfolgen. Das bedeutet, dass die Dartscheibe als Hintergrund eines Bildes dient und ein geworfener Pfeil den Vordergrund darstellt. Auch hierbei lassen sich 2 Bilder (vor dem Wurf und nach dem Wurf) miteinander vergleichen und die Unterschiede in einem Schwarz-Weiß-Bild darstellen. Danach wird die Pfeilorientierung anhand des Pixelverhältnisses zwischen der vorderen Pfeilspitzenregion und der hinteren Pfeilfedernregion bestimmt. Danach sollte sich der äußerste Punkt in der vorderen Region als der Trefferpunkt mit dem Dartboard definieren lassen.

Bei mehreren Pfeilwürfen hintereinander muss wiederholt ein „Hintergrundbild" mit bereits vorhandenen Pfeilen als Ausgangsbild aufgenommen werden, welches für einen weiteren Pfeilwurf als Referenz dient.

2.2 Ablauf des Analyse und Detektionsverfahrens

Es ist bisher davon auszugehen, dass eine Kalibrierungsphase für die Bildanalyse benötigt wird. In dieser sollten alle benötigten Analysemethoden auf die konkreten Rahmenbedingungen wie Licht, Schatten, Rauschen, Kameratyp, etc. abgestimmt werden. Diese Phase wird wahrscheinlich eine manuelle Einstellung durch den Benutzer erfordern. Danach kann das eigentliche Spiel beginnen, das heißt es wird zunächst die Spielvariante und die Spieleranzahl festgelegt. Nach dem Start des Spiels findet dann fortlaufend eine Bildanalyse statt, welche Pfeile erkennt, Punkte den Spielern zuordnet und die ggf. die Kalibrierung anpasst. Das Folgende Bild skizziert den groben Ablauf.

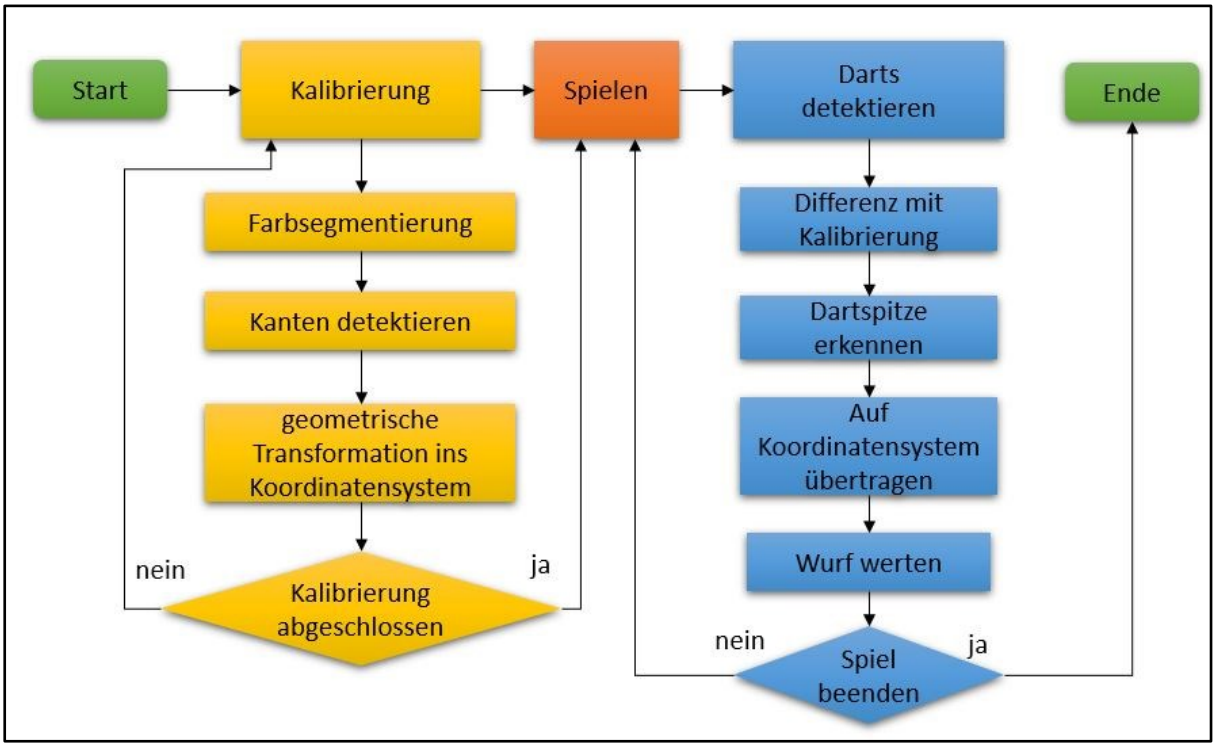

Abbildung 2 Theoretischer Programmablauf

3 PRAKTISCHE UMSETZUNG

Für die Umsetzung einer visuellen Bilderkennung wären viele verschiedene Kamera bzw. Computersysteme möglich gewesen. Das Team entschied sich aus pragmatischen Gründen dafür ein **Smartphone** als Plattform für die Umsetzung zu nutzen. Moderne Smartphones bieten bereits leistungsstarke Kameras und können auch genug Rechenleistung für die Bildanalysen mitbringen. Weiterhin sind diese fast allgegenwärtig verfügbar. Alle Teammitglieder besaßen Smartphones mit **Android** Betriebssystem weshalb sich hier auf diese Plattform beschränkt wurde.

Nach ausführlicher Recherche wurde sich für die Bildanalyse für die **OpenCV** Bibliothek entschieden. Diese ist frei verfügbar und bietet eine konkrete Unterstützung für die Android Plattform.

3.1 VORAUSSETZUNGEN

3.1.1 Softwareanforderungen

Ein Großteil der Softwareanforderungen finden sich in der entsprechenden *„build.gradle“* Datei im Projekt selbst. Dort sind die Abhängigkeiten beschrieben bezüglich Third-Party Bibliotheken, Plattform Version, etc. Dennoch seien an dieser Stelle die wesentlichsten Mindestvoraussetzungen genannt.

- Java 1.8
- Android SDK 28 (auch auf entsprechenden Testgerät)
- Android NDK 22
- CMake (gehört zu den Android Studio SDK Tools und kann darüber auch installiert werden)
- OpenCV Android SDK 4.5.1
- Android Studio 4.1.0
- Git (Nicht zwangsläufig nötig, jedoch für die kollaborative Arbeit eines der gängigsten Versionsverwaltungssysteme in der Softwareentwicklung)

3.1.2 Hardware- und physikalische Anforderungen

Die Hardwareanforderungen richtigen sich hierbei vor allem an die genutzten Smartphones. Zwar bietet Android Studio ein Simulationsmodus für Smartphones, dieser ist jedoch für einen Produktiven Einsatz ungeeignet, da dieser nur eingeschränkte Funktionalitäten bietet. Über die konkreten Hardwaremindestanforderungen lassen sich noch keine Aussagen treffen, da diese auch stark abhängig von der Implementierung der Software sind. Als Testgerät wurde ein **Galaxy S10+** verwendet, weil dieses zum aktuellen Entwicklungsstand vergleichsweise leistungsstark war. In der Entwicklung zeigte sich bereits, dass dieses durchaus in der Lage war die Bildanalysen durchzuführen.

Neben dem Smartphone Spielen die Umweltbedingungen ebenfalls eine Rolle. Folgende Empfehlungen können bereits gegeben werden:

- Ausleuchtung mit Licht im natürlichen Spektralbereich, um möglichst Farbverfälschungen zu vermeiden und die Farbsegmentierungen zu verbessern.
- Diffuse Ausleuchtung der Dartscheibe ohne Schattenwurf. Schatten verfälschen bzw. „verschieben“ die detektierten Ränder zwischen den Bereichen an den Metalldrähten
- Standsichere Kameraposition. Es ist essentiell, dass während der Detektionsphasen keinerlei Kamerabewegungen stattfinden. Hier bietet sich ein Smartphone-Stativ an einfache Möglichkeit an.

3.2 Quick Start Guide

Bevor auf die technischen Details des Projekts eingegangen wird, folgt eine kurze Erklärung wie der aktuelle Stand des Projekts gestartet werden kann.

Neben den Projektressourcen und den zu installierenden SDK Tools ist es notwendig das OpenCV Android SDK **separat** herunterzuladen und zu entpacken. Im Hauptverzeichnis des „**DartsTracker**"-Projekts befindet sich die Datei „*settings.gradle*". Dort muss der Pfad zum OpenCV SDK angepasst werden.

```
// change this path to your SDK
project(':opencv').projectDir = new File('D:/OpenCV-android-sdk/sdk')
```

Auszug 1 OpenCV SDK Pfad in sesstings.gradle

Erst danach kann das „**DartsTracker**"-Projekt kompiliert werden.

Dann kann bereits ein virtuelles Smartphone über den **AVD Manager** eingerichtet oder vorhandenes Testgerät genutzt werden.

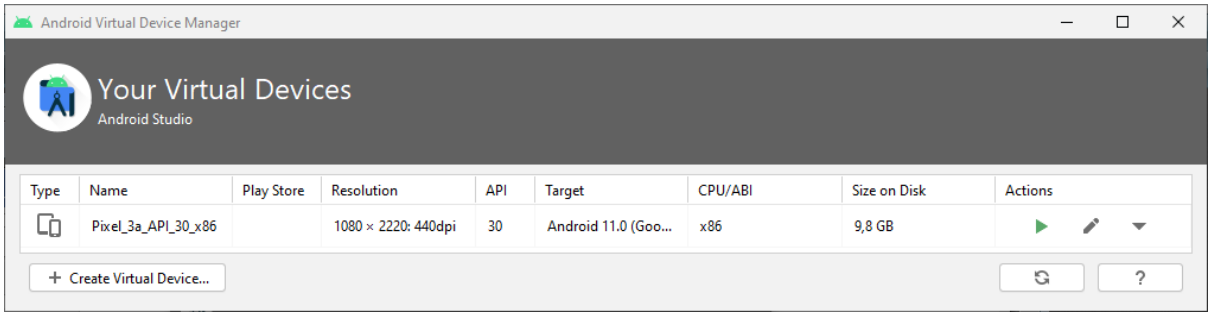

Abbildung 3 AVD Manager

3.3 Applikationsstruktur

3.3.1 Module

Die Projektstruktur gliedert sich in 2 Module auf.

- „**dartstracker**" – enthält die Android App und alle dazu benötigten Ressourcen
- „**opencv**" – enthält die komplette OpenCV Android SDK

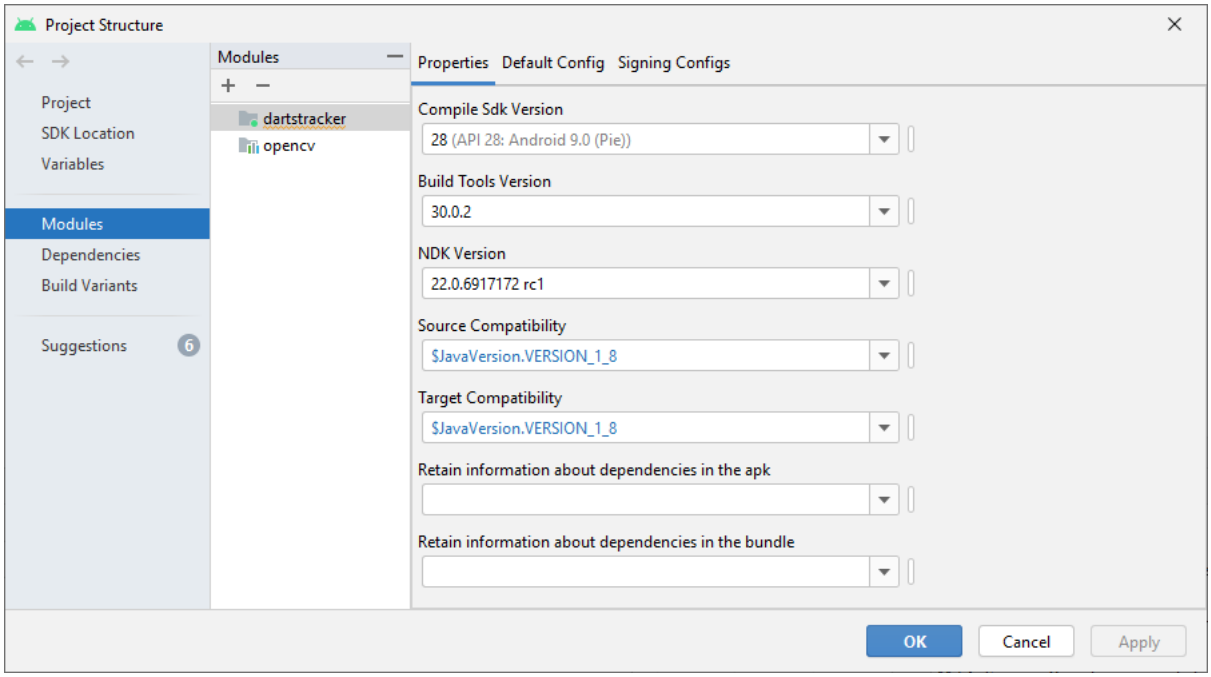

Abbildung 4 Projektmodule

3.3.2 Modul dartstracker

Die Struktur innerhalb des Dartstracker orientiert sich weitestgehend an den üblichen Standards für Android Projekte. Hier seien noch mal die wichtigsten Orte genannt:

Pfad	Beschreibung / Inhalt
dartstracker\src\main\java	Enthält eigene Java Sourcen
dartstracker\src\main\cpp	Enthält eigene C++ Sourcen
dartstracker\src\main\res	Enthält alle für Androidsapps üblichen Design und Textressourcen
dartstracker\src\main\AndroidManifest.xml	Metainformationen zur Applikation, Berechtigungen, etc.
dartstracker\src\test dartstracker\src\androidTest	Enthält Java und Android Unit Tests. Unter den Java Tests befinden sich auch nützliche Tests welche einzelne Funktionen der verwendeten Bilderkennung widerspiegeln ohne jedoch ein gesamtes Android Betriebssystem hochfahren zu müssen.
dartstracker\build.gradle	Wichtigste „Build"- Anleitung für den DartsTracker. Enthält Beschreibung sämtlicher Abhängigkeiten sowie externe Builds

Tabelle 1 Inhalte DartsTracker Struktur

Neben den applikationsspezifischen Dateien gibt es noch eine Reihe von rudimentär vorhandenen Continuous Integration bzw. Continuous Delivery Funktionen. Dazu gehört unter anderem **„Fastlane", „gitlab-ci-deactivate.yml", „Dockerfile"**, etc. Diese kam jedoch zur Entwicklungszeit nie zum Einsatz und sind daher nicht abschließend eingerichtet bzw. gänzlich deaktiviert worden.

3.3.3 Aufgaben des Sourcecodes

Um die benötigten Funktionen zu kapseln, wurde eine Reihe von Java sowie einer C++ Klasse erstellt. Für die Oberflächendesigns existieren – für Android Apps typisch – eine Reihe von Design-XMLs in folgenden Ordnern:

- res/drawable – Icons
- res/layout – Activity Designs
- res/menu – Menu Designs
- res/mipmap – App Icons für das Android Betriebssystem
- res/values – Texte und Styles

Die folgende Übersicht beschreibt nur im Überblick die generellen Aufgaben der Java / C++ Klassen und deren Zusammenhänge.

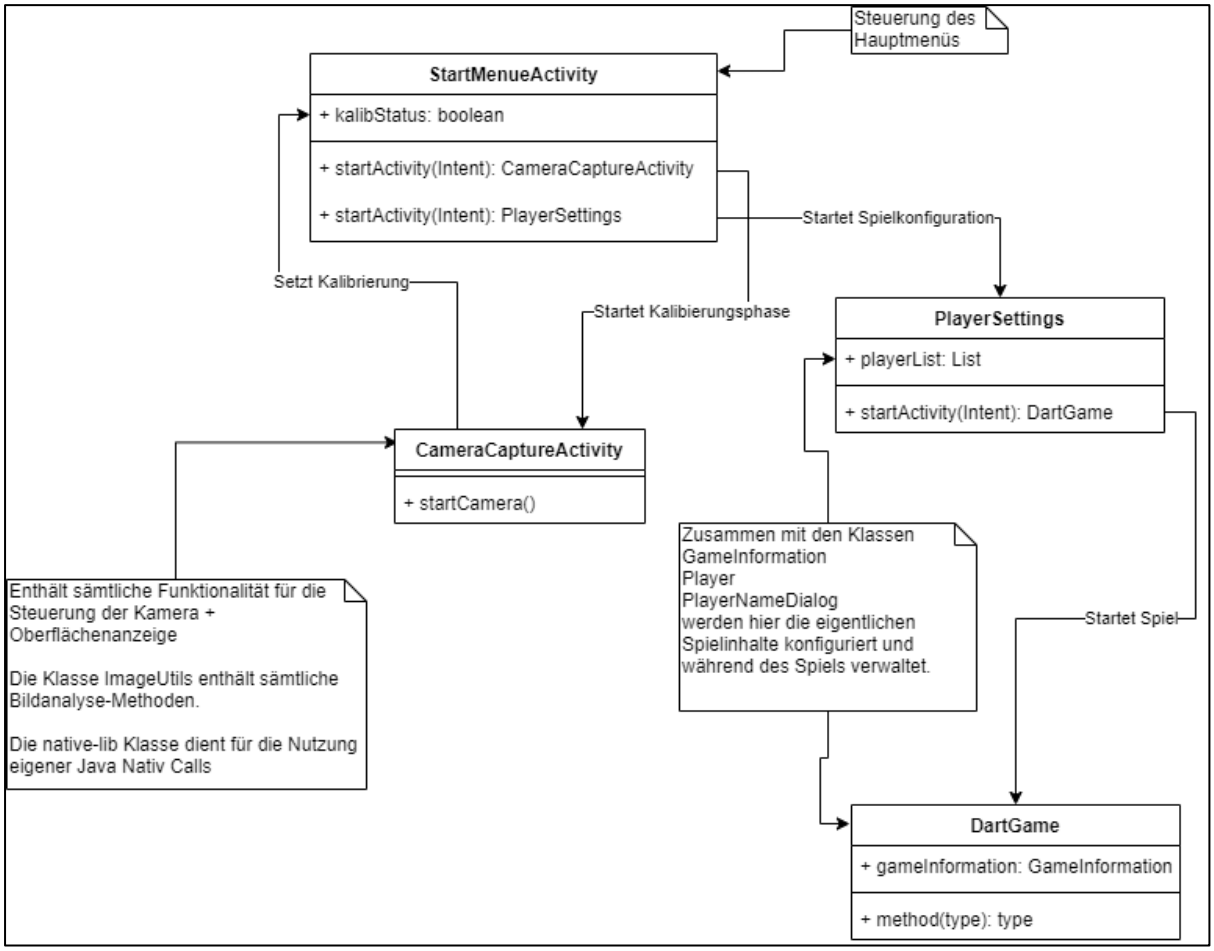

Abbildung 5 Strukturelle Übersicht der Klassen

3.4 Oberfläche

Die Applikation besteht aus vier Abschnitten, die jeweils eigene Funktionalitäten implementieren. Zu diesen vier Abschnitten gehören das Hauptmenü, der Kalibrierungsabschnitt, das Einstellungsmenü und die Spieloberfläche.

3.4.1 Hauptmenü

Das Hauptmenü bildet den Anfang der Applikation. Der Nutzer hat die Möglichkeit das Einstellungsmenü eines Dart Spieles (*Dart – Game – Start*), den Kalibrierungsabschnitt zu öffnen (*Kalibrierung*) oder die Applikation zu beenden (*Anwendung beenden*).

Abbildung 2 Hauptmenü der Applikation

Die einzelnen Funktionalitäten wurden mit Hilfe der „CardViews“ der „AndroidX“ – Bibliothek umgesetzt. Darin wurden Textfelder eingerichtet, die den nächsten Abschnitt nennen oder die Funktionsweise der „CardView“ näher erläutern sollen.

Darüber hinaus wurde im Hauptmenü implementiert, dass die Applikation über die „CardView“ *Dart – Game – Start* nur fortgesetzt werden kann, damit der Nutzer der Schritt *Kalibrierung*“ nicht überspringt. Weiterhin wurde ein Textfeld eingefügt, das den Nutzer daraufhin weist, dass eine vorherige Kalibrierung vor dem eigentlichen Spielstart vollzogen werden muss.

3.4.2 Kalibrierung

Die Aufgabe der Kalibrierung ist es, ein Referenzbild aufzunehmen und eine Anpassung/Überprüfung der Bilderkennung (Licht, Schatten, Farben, etc.) vorzunehmen. Zu diesem Zweck stehen dem Nutzer die verschiedene Analysenverfahren einzeln zur Verfügung. Deren Liste kann über ein Menüsymbol in der rechten oberen Ecke aufgerufen werden. Die Kalibrierungsansicht teilt sich hierbei in 2 Bereiche.

Der Obere Bereich zeigt das aktuell aufgenommene Kamerabild. Im unteren Bereich ist Ausgabe bei Nutzung einer Bildanalyse zu sehen.

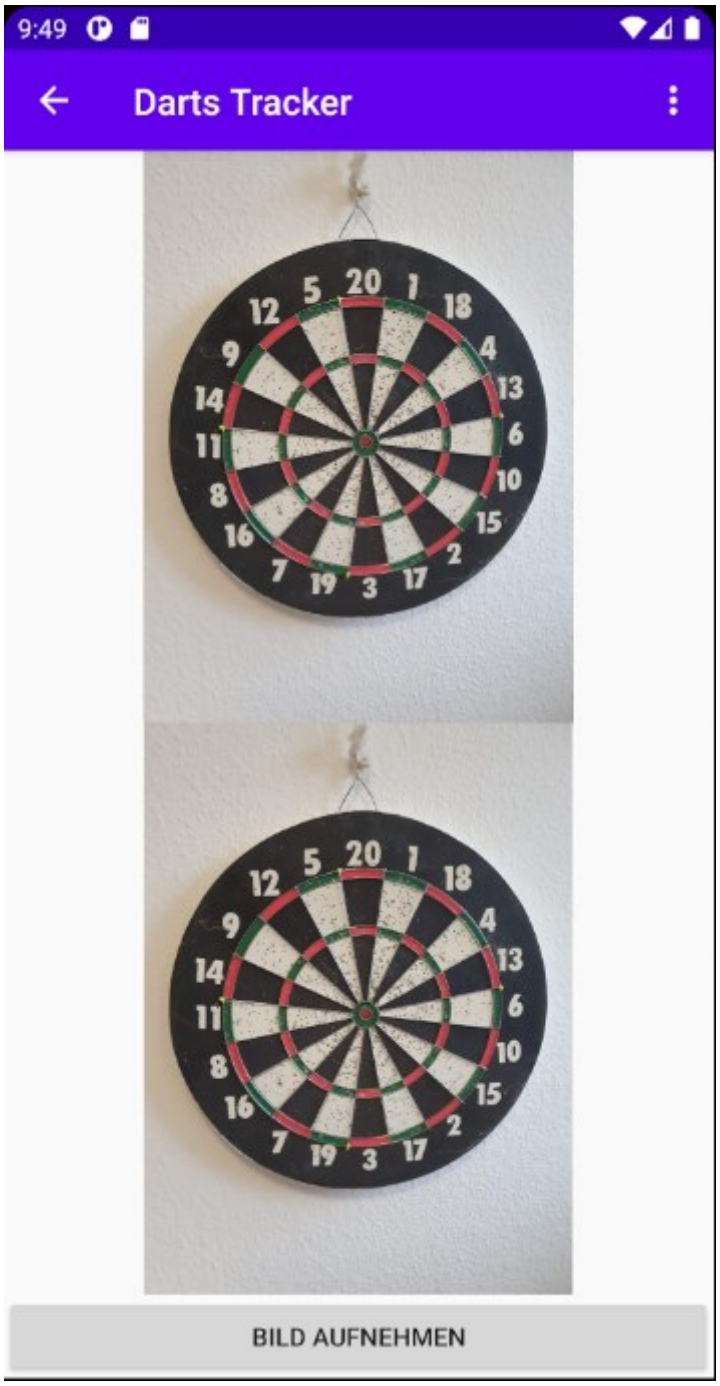

Abbildung 3 Kalibrierungsabschnitt der Applikation

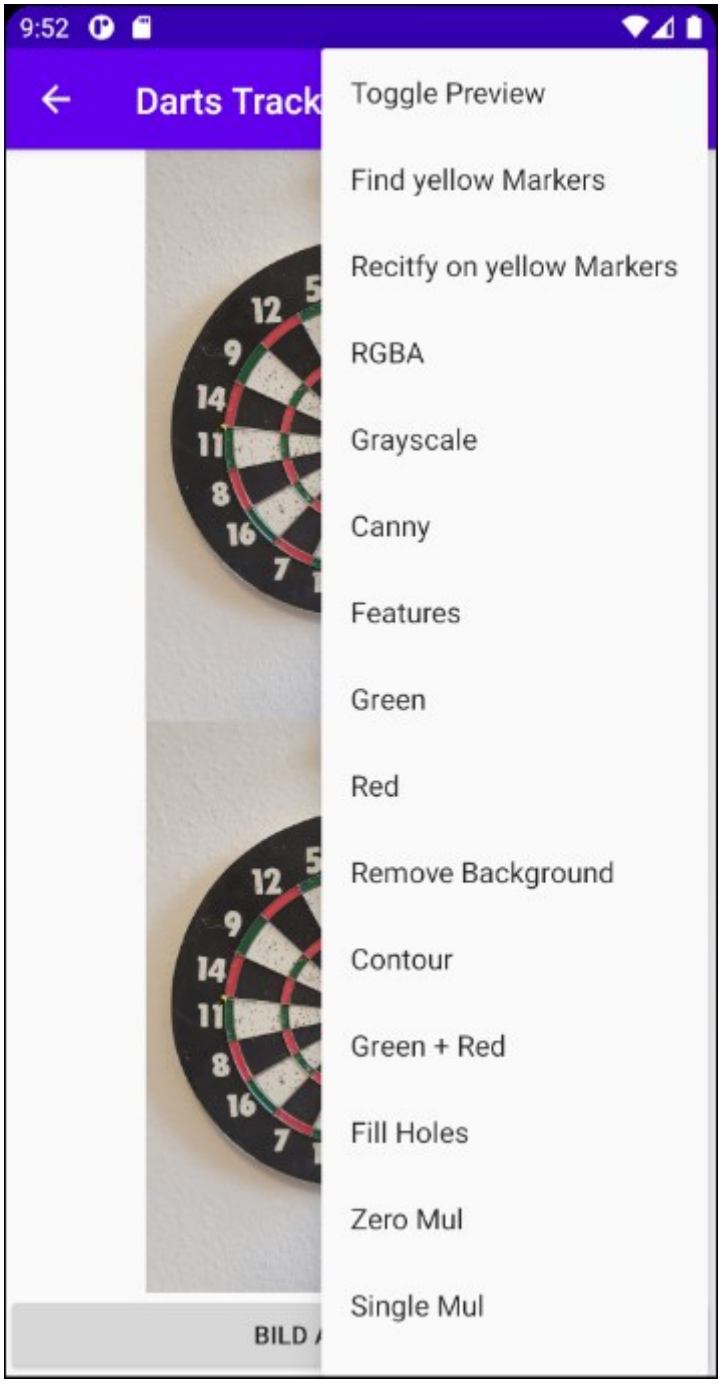

Abbildung 4 Kalibrierungsabschnitt mit Analysefunktionen

Sofern eines der Analyseverfahren ausgewählt wird, ändert sich das untere der beiden Bilder.

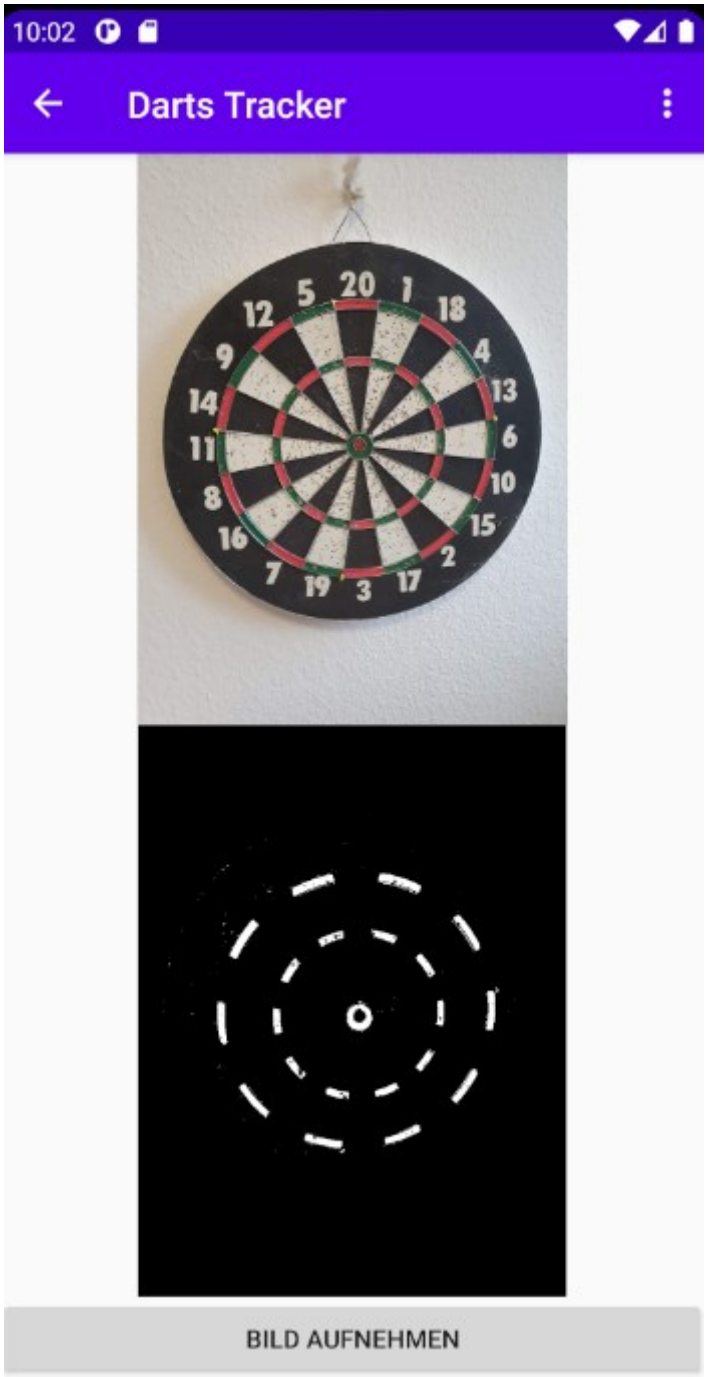

Abbildung 4 Beispiel Detektion „Green“

Der „Button“ am unteren Bildschirmrand speichert die gesammelten Informationen lokal in dem Speicher des Mobilgerätes. Nach dem Speichervorgang, wird der Nutzer ins Hauptmenü zurückgeführt, in dem nun die Möglichkeit zur Fortführung der Applikation, *Dart – Game – Start*, eröffnet wurde.

3.4.3 Einstellungsmenü

Das Einstellungsmenü übernimmt die Aufgabe verschiedene Einstellungen für das anstehende Spiel anbieten zu können. Der Nutzer hat hierbei die Möglichkeit die Spieleranzahl zu erhöhen und im Anschluss daran den Dart – Taschenrechner zu starten. (*Weiter*)

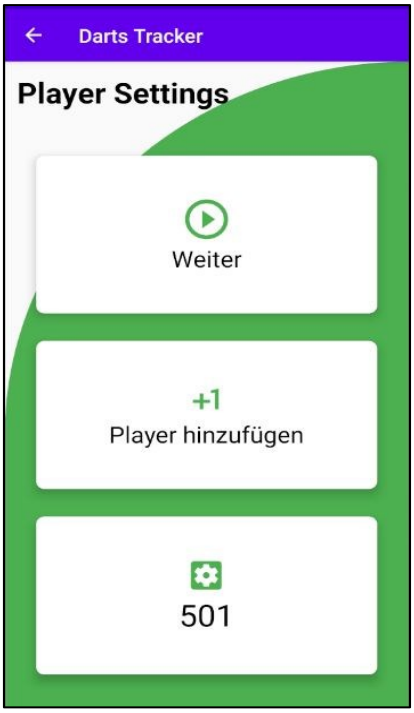

Abbildung 7 Einstellungsmenü der Applikation

Sofern ein neuer Spieler dem Spiel hinzugefügt werden soll, öffnet sich ein Eingabefenster, dass den Spielernamen des neuen Spielers erwartet.

Abbildung 8 Eingabe des Spielernamens

Mit der Bestätigung des Spielernamens, *Spielernamen bestätigen*, wird ein neues Spielerobjekt angelegt, dass wesentlich Informationen über den Spieler hält. Darüber hinaus wird dem Nutzer der Applikation unterhalb der Textanzeige: *Player hinzufügen*, der hinzugefügte Spieler in Form seines Namens repräsentiert.

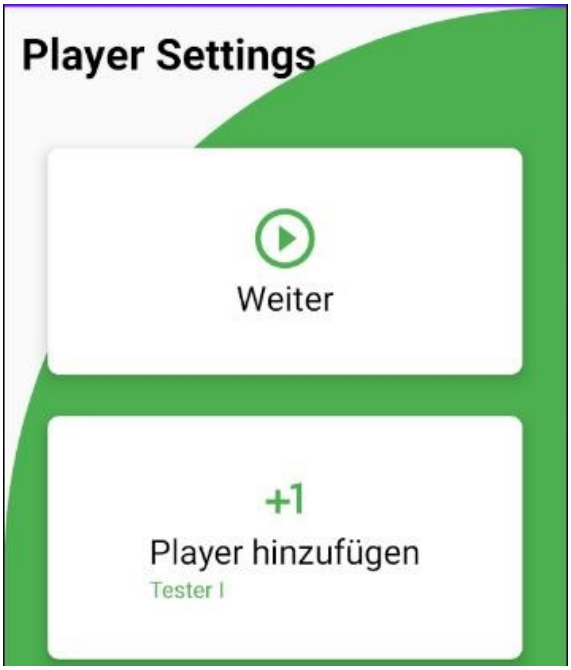

Abbildung 9 Spielererstellung

3.4.4 Dart – Taschenrechner

Nachdem ein Spieler angelegt wurde und der Nutzer den *Weiter* – Button verwendet hat, öffnet sich der Dart – Taschenrechner. Dieser besitzt die Aufgabe die Gesamtpunktzahl jedes Spielers zu bestimmen und die geworfenen Punkte durch jeden Dart anzuzeigen und von der Gesamtpunktzahl abzuziehen.

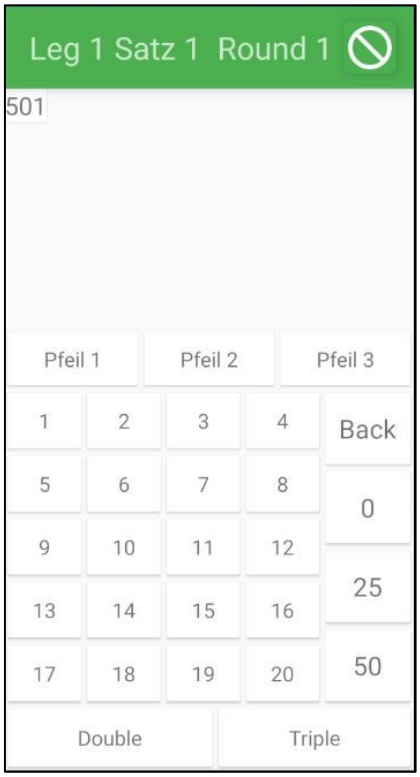

Abbildung 10 Dart - Taschenrechner ohne Punktabzug

Zum aktuellen Entwicklungsstand des Projekts, hat der Nutzer die Möglichkeit manuell über das Taschenrechnersystem die Punktzahl jedes Darts einzutragen. Nach der Eingabe der geworfenen Punktzahl mit Hilfe des ersten Darts, ändert sich auch die Anzeige *Pfeil 1* zur gewünschten Punktzahl, beispielsweise 20.

Abbildung 11 Dart - Taschenrechner mit Punktabzug (eigene Abbildung)

Des Weiteren wurden die Felder „Double“ und „Triple“ in das Taschenrechnersystem eingebaut, damit der Nutzer das Treffen der „Double“ oder „Triple“ – Felder auf dem Board simulieren kann. Die letzte Funktionalität, die implementiert wurde, wird durch den Button *Back* ausgelöst. Hierbei handelt es sich um die Zurücknahme des letzten Darts, falls eine fehlerhafte Zahl eingetragen wurde.

4 Ergebnisse und Stand des Projekts

Um die Organisation und Planung der Gruppenarbeit zu erleichtern, wurden Teilaufgaben definiert und diese nach Art von Projektmeilensteinen grob festgelegt. Anhand derer lässt sich der Gesamtfortschritt des Projekts besser nachvollziehen. Die nachfolgende Auflistung beschreibt die einzelnen Aufgabenschritte bis zu Fertigstellung der App und ob dieses Ziel bereits umgesetzt wurde.

Meilenstein/Aufgabe	Erledigt
Erkennung des Dart Boards	
Erkennung von Grün-Flächen	✓
Erkennung von Rot -Flächen	✓
Erkennung der Metallkanten durch „Canny“-Algorithmus	✓
Erkennung Bull’s Eye und Bull	✓
Perspektivische Entzerrung der Aufnahmen	✓
Transformation der „Dart“ – Board – Pixel in Koordinatensystem	✗
Speicherung der tatsächlichen Kalibrierung	✗
Erkennung der Dartpfeile	
Video-Stream des Dart Boards	✓
Automatischer Vergleich mit kalibriertem Bild	✗
Erkennung der Pfeile durch Differenz der Bilder	✗
Identifizierung der Punkte (Lokalisierung der Pfeilspitze im Koordinatensystem)	✗
Erstellung einer App für die Darstellung der Ergebnisse	
Oberfläche für die Kalibrierung des Referenzbildes	✓
Oberfläche für Spieleinstellungen	
Spieleranzahl	✓
Punkte pro „Leg“	✗
Oberfläche für die Zählung der Spielerpunkte	
„Taschenrechner“-System für manuelle Eingaben	✓
Automatische Errechnung der Punkte durch Erkennung der Pfeile	✗

Tabelle 2 Projektstand

Die Übersicht stellt dar, dass in den Bereichen der Erkennung der Dartpfeile sowie in der Erstellung der Applikation selbst verschiedene Umsetzungen fertig gestellt worden sind. Zusammenfassend ist festzustellen, dass im Rahmen des Modules „Teamprojekt“ verschiedene Bildanalyse-Algorithmen bereits implementiert werden konnten, sowie einen einen manuellen Punktezähler innerhalb der Applikation.

5 PROJEKTFORTFÜHRUNG

Sofern eine Fortführung des Projekts durch die Gruppe selber oder durch andere Interessierte gewünscht ist, finden sich folgende Vorschläg, um die Arbeit an dem Projekt zu verbessern bzw. Lösungsideen zu noch nicht umgesetzten Anforderungen.

- OpenCV Modulabhängigkeit entfernen und native OpenCV Libs direkt ins „DartsTracker“-Projekt inkludieren. Die OpenCV SDK ist sehr groß und liefert auch sämtliche Sourcen zum selbst Kompilieren. Bislang werden lediglich die konkreten nativen Android Bibliotheken benötigt. Eine Direkte Einbindung würde den Build-Lifecycle erheblich verschlanken.
- In der Kalibrierungsphase wird es wahrscheinlich notwendig sein, dass der Nutzer selbst an den Schwellwerten der Farben oder an der Rauschunterdrückung Änderungen vornehmen kann, weil diese stark abhängig sind von Licht und Schatten der Umgebung und der verwendeten Kamera. Hier sollten entsprechende Slider implementiert werden zu den jeweiligen Analysealgorithmen.
- Stärke Nutzung von Java Unit-Tests. Es hat sich im letzten Drittel der Bearbeitungszeit gezeigt, dass eine Nutzung solcher Tests die eigentliche Entwicklungszeit beschleunigen kann, weil konkrete Bildanalyseprobleme unabhängig der (schwerfälligen) Android Projektstruktur gelöst werden können.
- Optimierung der Bildanalysen hinsichtlich der Performance. Die Gesamtperformance ist stark abhängig von der Anzahl/Geschwindigkeit/Größe der aufgenommenen Frames. Ist sollte hier ebenso Einstellungen für den Nutzer geben um diese Analyse an sein jeweiliges Gerät anpassen zu können. Echtzeiterkennung von Pfeilwürfen ist für Nutzer auch gegeben, wenn diese beispielsweise nur alle 2 Sekunden passiert.
- Die Arbeitsbereiche zwischen den Projektmitgliedern sollten sich möglichst gut voneinander kapseln lassen und dies sollte sich auch im Quellcode widerspiegeln. Während des Projekts kam es zu parallelen Arbeiten an gleichen Stellen im Quellcode, was später einen erhöhten Merge-Aufwand bedeutete. Es hat sich als nützlich herausgestellt, wenn Arbeiten klarer getrennt werden wie z.B.: Erstellung der Oberfläche, Lösen der Bildanalysen in Unit-Tests, Implementation der Lösungen in die Applikation.

6 FAZIT

Wie aus dem Projektstand erkennbar ist, konnten die festgelegten Meilensteine zu Beginn des Modules nur zum Teil umgesetzt werden.

Dieser Teilerfolg ist darauf zurückzuführen, dass die Gruppe zuvor weder Kenntnisse in der Programmierung von Android-Applikationen noch in der Bibliothek „OpenCV“ besaß. Aus diesem Grund heraus bestand für die Gruppe ein erhöhter Schwierigkeitsgrad in der Einarbeitung der technischen Werkzeuge.

Der zweite Grund liegt in der kurzen Bearbeitungszeit. Aufgrund der Komplexität der Problematik, sowie der zuvor genannten schwachen Vorkenntnisse blieb eine lange Vorbereitung – und Einarbeitungsphase nicht aus, weshalb die eigentliche Umsetzungszeit erst zu einem späteren Zeitpunkt beginnen konnte. Darüber hinaus musste aufgrund der damaligen COV19 – Lage bereits ein verkürztes sowie reines Online Semester in Kauf genommen werden.

Dennoch konnten aus der anspruchsvollen Aufgabenstellung viele gute Ergebnisse mitgenommen werden. Darunter gehört die Erfahrung in verschiedene Algorithmen für die Auswertung von Bildern, das Sammeln an Erfahrungen in der Implementierung einer „Android App“ sowie die Vertiefung in die Oberflächenprogrammierung.

Darüber hinaus gehörte die gute Kommunikation zwischen den Studierenden, aber auch zum Betreuer selbst, in Form von wöchentlichen Meetings zu den Stärken dieses Modules. Unklarheiten blieben dadurch nie lange bestehen und es konnte kontinuierlich an dem Projekt gearbeitet werden.

Zusammenfassend kann das erreichte Ergebnis dieses Projektes als eine Kombination aus Komplexität, erschwerten Bedingungen und sehr guter Kommunikation zwischen Studierenden und dem Betreuer betrachtet werden. An dieser Stelle spricht die Projektgruppe eine klare Empfehlung für die Fortführung des Projekts aus.