Architekturentwicklung eines Hochschulinformationssystems

INFM2300– Teamprojekt – Sommersemester 2021

Autoren:

Andreas Voigt, *andreas.voigt@hochschule-stralsund.de*, Mat.Nr.: 19398

Ricardo Lissner, *ricardo.lissner@hochschule-stralsund.de*, Mat.Nr.: 19134

Hannes Hoffmann, *hannes.hoffmann@hochschule-stralsund.de*, Mat.Nr.: 17033

Eingereicht am: 04.08.2021

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.

Inhaltsverzeichnis

1 Aufgabenstellung

Die Aufgabenstellung des Teams „Darts Tracker“ bestand darin, dass anhand von Videoaufnahmen eines Dartboards, Trefferpunkte durch Dartpfeile analysiert und dem richtigen Spieler angerechnet werden. Diese Aufgabe ist eine Fortführung aus einer früheren Arbeit, welche bereits die Grundsteine für die Erstellung einer Lösung gelegt hat.

In der Fortführung dieser Aufgabe wurden weitere Aspekte einbezogen wie die regelgerechte Wertung von Treffern. Dabei sollten mindestens die Spielvarianten 301/501 mit vereinfachtem Single-Out implementiert werden. Weiterhin sollen die Erkennung und Auswertung in Echtzeit, also während des Spielens erfolgen. Der jeweilige Spieler erhält damit direkt eine Rückmeldung über den Treffer und seine erreichten Punkte. Bei Fehlwürfen oder falsch erkannten Würfen soll es eine Möglichkeit einer manuellen Eingabe bzw. Korrektur der Würfe geben. Ebenfalls soll nun ein Ende des Spiels inklusive Darstellung des Siegers erfolgen.

Die hier in den folgenden Kapiteln beschriebene technische Lösung stellt einen so genannten **Proof of Concept** (PoC) dar. Das heißt es wird die prinzipielle Durchführbarkeit anhand eines Prototyps gezeigt. Die wesentlichsten Kernanforderungen werden dabei erfüllt, ohne den Anspruch auf vollständige Richtigkeit zu erheben. Die Ergebnisse und die Grenzen dieses PoC werden in den Kapiteln 6 und 7 näher betrachtet.

Zusammengefasst stehen folgende Aufgabenpunkte dabei im Vordergrund:

1. Erkennung des Dartboards
2. Möglichkeit zur Kamerakalibrierung
3. Trefferauswertung
    a. Erkennung eines Dartpfeils
    b. Ermittlung der Pfeilspitze
    c. Auswertung der geworfenen Punktzahl
4. Ausschluss bereits vorhandener Treffer
    a. “Alte Treffer” werden nicht berücksichtigt
    b. Doppeltreffer sind zu vernachlässigen
5. Live-Betrieb

2 Technische Voraussetzungen

Als Voraussetzung für die Aufnahme und Live-Auswertung von Bildern wäre mindestens ein Kamerasystem und ein angeschlossener Computer nötig. Ein Smartphone als Plattform bietet bereits alles innerhalb eines Gerätes. Moderne Geräte sind zudem leistungsstark genug, um Bilder in Echtzeit zu analysieren. Auch die hohe Mobilität und allgegenwärtige Verfügbarkeit spricht für die Nutzung eines Smartphones. Alle Teammitglieder besaßen Smartphones mit Android Betriebssystem weshalb sich hier auf diese Plattform festgelegt wurde.

Für die Arbeit in der Android OS Umgebung bietet Android Studio ein Simulationsmodus für Smartphones, dieser ist jedoch für einen produktiven Einsatz ungeeignet, da dieser nur eingeschränkte Funktionalitäten bietet.

Nach einer ausführlichen Recherche wurde sich für die OpenCV Bibliothek entschieden um die Bildanalysen durchzuführen. Diese ist frei verfügbar und bietet eine umfassende Unterstützung für die Android Plattform.

2.1 Hardware- und Umweltvorrausetzungen

Die Hardware- und Umweltanforderungen richtigen sich an insgesamt 3 Aspekte – an die genutzten Smartphones, die Spielmaterialien und an die Umweltbedingungen.

Für die Entwicklung lagen nur die 3 Testgeräte des Teams vor. Auch auf dem vergleichsweisen hardwareschwächsten Modell konnte die Applikation betrieben werden, weshalb dieses als Mindestanforderung herangezogen wird. Es ist jedoch davon auszugehen, dass auch schwächere Hardware ausreichend ist, da das genutzte System nicht an der Belastungsgrenze arbeitete.

Im Folgenden sind die wesentlichsten Smartphone-Anforderungen genannt:

- **Snapdragon 630** (Octa-Core 2,2Ghz)
- **150 MB freier Speicher**
- **130 MB RAM**
- **Rückkamera** (alle heutzutage üblich verbauten Kamerasysteme sind ausreichend leistungsstark)

Für die Spielmaterialien wurden Vorgaben getroffen, damit die Applikation korrekt arbeiten kann:

- Das Dartboard muss nach dem geltenden internationalen Reglement[1] die **richtigen Abmessungen besitzen**. (Das schließt auch die Drahtstärke ein.)

[1] Nachzulesen unter https://de.wikipedia.org/wiki/Darts#Abmessungen_der_Dartscheibe

- **4 gelbe Marker** müssen auf dem Board korrekt angebracht werden. (mehr dazu im Kapitel 3 Technische Lösung)
- Die genutzten **Pfeile müssen komplett einfarbig blau** sein
- Die **Farben auf dem Board** müssen klar als **Schwarz, Weiß, Grün und Rot** erkennbar sein. Keine beigefarbenen Segmente wie sie bei vielen Kork-Boards zum Einsatz kommen.
- Der **Bildhintergrund sollte frei von Störeinflüssen sein** wie farbige Tapeten, Gegenständen, etc. Eine weiße Wand ist ideal.

Umweltbedingungen verbessern die Erkennungsleistung der Bildanalyen. Folgende Empfehlungen können ausgesprochen werden:

- Ausleuchtung mit **Licht im „warmen" bis natürlichen Spektralbereich**, um möglichst Farbverfälschungen zu vermeiden und die Farbsegmentierungen zu verbessern. (ca. 2700K – 5500K bei LED-Leuchtmitteln)
- **Vermeidung von starkem Schattenwurf** auf dem Board, um Farbübergänge leichter erkennbar zu machen.
- Eine **standsichere Kamera**, z. B. mit Hilfe eines Stativs ist für ein bequemes Spielen ohne Nachjustieren anzuraten

2.2 Software

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** (ab Android Version 9.0 – auch bekannt als Android Pie.) Auch das später genutzte Smartphone muss diese Android Version unterstützen.
- **Android NDK 22**
- **CMake** (gehört zu den Android Studio SDK Tools und kann darüber auch installiert werden)
- **OpenCV Android SDK 4.5.2**
- **Android Studio 4.2.1**
- **Git** (Nicht zwangsläufig nötig. Für das vorliegende Projekt wurde es jedoch als kollaboratives Versionsverwaltung genutzt)
- **Maven ab Version 3.3** (nur für das gesonderte Entwicklungs- und Testprojekt)
- **Python ab Version 3.8** (nur für das gesonderte Entwicklungs- und Testprojekt)

2.3 Entwicklungsumgebung

Android Studio ist eine sehr umfangreiche und ausgereifte IDE für die Entwicklung von Apps, dennoch hat sich in der praktischen Umsetzung die Arbeit direkt auf dem Android OS im Zusammenhang mit OpenCV als umständlich und mühevoll herausgestellt. Es wurde daher ein gesondertes Java Maven Projekt mit Namen *poc* geschaffen, sowie separate *Python Skripte*. Beides befindet sich ebenfalls im Gesamtprojekt. Wie diese genutzt werden können, wird im Kapitel 2.4 Quickstart kurz erläutert.

In diesem Java Maven Projekt bzw. den Skripten lassen sich die zu entwickelnden OpenCV Bildanalyse Algorithmen deutlich leichter und schneller entwickeln, da diese von der Android Systemumgebung vollständig entkoppelt wurden. Je nach Präferenzen des Entwicklers können diese Hilfsprojekte genutzt werden, wobei das Java Projekt bereits eine nahezu 1 zu 1 Übernahme in die Android Umgebung zulässt. Die darin enthaltenen Lösungen bilden Teile der Gesamtlösung ab oder enthalten noch verworfene Lösungsansätze.

2.4 Quickstart

2.4.1 Darts Tracker

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 *(eigener Auszug)*

Erst danach kann das „Darts Tracker“-Projekt kompiliert werden.

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

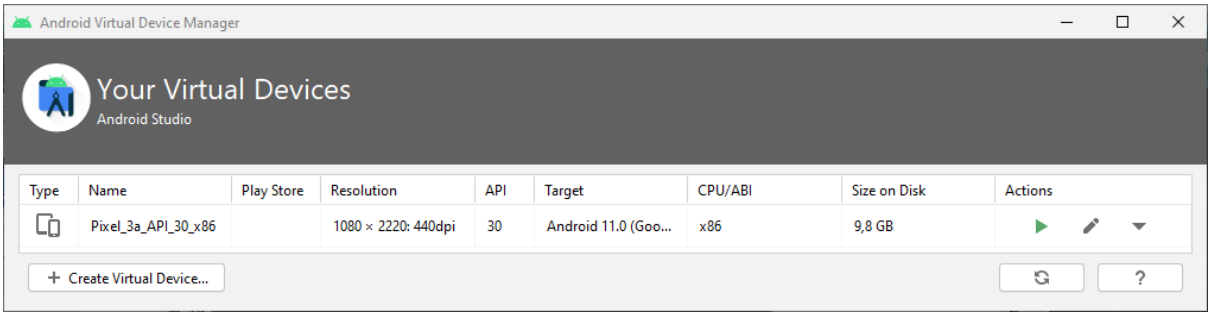

Abbildung 1 – AVD Manager *(eigene Abbildung)*

2.4.2 Java PoC

Das separate OpenCV Entwicklungsprojekt *poc* befindet sich im gleichnamigen Ordner *poc*. Es ist ein Standard Java Maven Projekt und kann mit einer IDE der Wahl geöffnet werden. Da OpenCV plattformabhängig ist, muss ggf. entsprechende Library zur Verfügung stehen[2] und diese in den Java Quellen entsprechend genutzt werden. Als Beispiel ist bereits die OpenCV Java Bibliothek für Windows x64 Systeme integriert im Pfad:

poc/src/main/resources/opencv_java452_windowsx64.dll

Diese wird in den Java Quellen wie folgt eingebunden (am Beispiel BlueDartTracking.java):

```
static {
// programatically load opencv native lib [...]
System.load(BlueDartTracking.class.getClassLoader().getResource("opencv_java452_windowsx64.dll").getPath());
}
```

Auszug 2 – Einbindung OpenCV Java Bibliothek (eigener Auszug)

Die Nutzung dieses Projekts ist optional. Es beeinflusst in keiner Weise das Android Projekt und kommt auch nicht auf Smartphone zum Einsatz.

2.4.3 Python Skript

Für die Entwicklung von Algorithmen stehen auch separate *Python Skripte* zur Verfügung. Diese befinden sich im Pfad:

dartstracker/src/test/resources/scripts

Für die Nutzung kann mit einer beliebigen IDE gearbeitet werden. Die entsprechende OpenCV Python Bibliothek (und deren Abhängigkeiten) muss jedoch zuvor installiert sein. Es ist hilfreich hier ein Paketmanager wie *pip* einzusetzen, so dass die Bibliothek wie folgt installiert werden kann:

```
pip install opencv-python
```

Auszug 3 – Einbindung OpenCV Python Bibliothek (eigener Auszug)

Die Nutzung dieser Skripte ist optional. Es beeinflusst in keiner Weise das Android Projekt und kommt auch nicht auf Smartphone zum Einsatz.

[2] Pakete können hier heruntergeladen werden: https://opencv.org/releases/

3 Technische Lösung

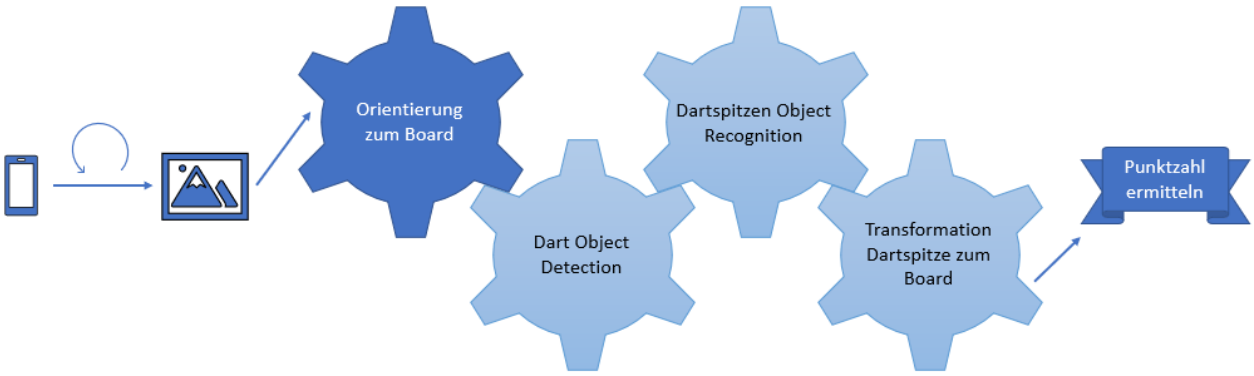

Abbildung 2 – Technischer Ablauf *(eigene Abbildung)*

Um zu einem geworfenen Dart eine Punktzahl zu ermitteln, müssen die aufgenommenen Frames verschiedene Phasen der Verarbeitung und Datenextraktion durchlaufen. Jede der Phasen adressiert hierbei eine andere technische Herausforderung, welche überwunden werden muss, um die entsprechende Punktzahl richtig bestimmen zu können. Die Abbildung 2 – Technischer Ablauf zeigt die durchlaufenen Phasen auf, welche im Folgenden etwas genauer betrachtet werden sollen.

3.1 Orientierung zum Board

Abbildung 3 – Referenzpunkte *(eigene Abbildung)*

Um herauszufinden, wie der Dartpfeil auf dem Dartboard auftrifft, muss zunächst ermittelt werden, wie das Dartboard in den Frames positioniert und ausgerichtet ist. Wie in Abbildung 3 – Referenzpunkte zu sehen ist, wurde sich hierbei dafür entschlossen konkrete

Punkte farblich hervorzuheben. Auf diese Weise kann die Orientierung zum Board immer eindeutig bestimmt werden, solange die Punkte gefunden werden.

Abbildung 4 – Farbsegmentierung auf Referenzpunkte (eigene Abbildung)

Wie Abbildung 4 – Farbsegmentierung auf Referenzpunkte zu sehen ist werden die hervorgehobenen Punkte dann über eine Farbsegmentierung selektiert, wobei die Schwerpunkte der resultierenden weißen Pixelmassen als Referenzpunkte identifiziert werden.

3.2 Dart Object Detection

Abbildung 5 – Dart Object Recognition (eigene Abbildung)

Das nächste Problem besteht darin den Dartpfeil als Objekt zu identifizieren. Auch hierbei wird sie die Farbsegmentierung zunutze gemacht, welche auf dem farbigen Dartpfeil selektiert. Auf diese Weise kann recht einfach eine gegen Null gehende Fehlerquote erreicht werden. Die

Abbildung 5 – Dart Object Recognition zeigt das Resultat, bei dem die Umrisse des Pfeils bereits deutlich zu erkennen sind. Jedoch können auch kleinere Pixelmassen gefunden werden, welche eindeutig nicht zum Dartpfeil gehören. Diese werden dezimiert indem kleiner Pixelmassen rausgefiltert werden. Um jedoch zu verhindern, dass potenziell Teile des Dartpfeils weggeschnitten werden, werden über einen Dilate-Erode-Algorithmus die Lücken zwischen nahen Pixelmassen zuvor geschlossen.

3.3 Dartspitze Object Recognition

Im letzten Kapitel wurde betrachtet, wie der Dartpfeil gefunden wird. Der Anwendung ist jedoch nicht bewusst, dass dies wirklich ein Dartpfeil ist oder in welche Richtung dieser ausgerichtet ist. Es ist lediglich eine Masse an Pixeln bekannt, welche bislang separiert wurden. Nun muss jedoch noch herausgefunden werden, wo der Dartpfeil auf das Dartboard trifft. Hierbei wird der Einfachheit halber nicht der komplette Dartpfeil erkannt, sondern lediglich die Spitze dessen, da es diese ist, welche auf das Dartboard trifft.

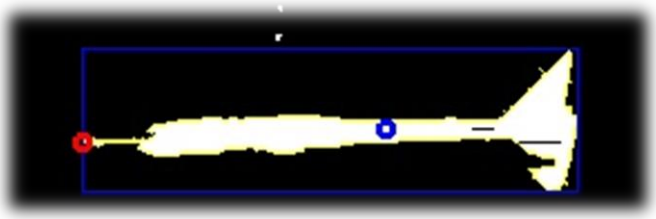

Abbildung 6 – Dartspitze Object Recognition (eigene Abbildung)

Hierfür wurde ein Algorithmus entwickelt, welcher zu der gegebenen Pixelmasse des Dartpfeils die Dartspitze identifiziert. Dafür wird in zunächst der Pixelschwerpunkt bestimmt. Der Pixelschwerpunkt ist der in Abbildung 6 – Dartspitze Object Recognition blau markierte Punkt. Er wird aus dem Mittelwert der Pixelkoordinaten gebildet, wobei lediglich die am Rand befindlichen Pixel verwendet werden, um Rechenzeit einzusparen. Anschließend wird der Pixel ausfindig gemacht, welcher sich am weitesten entfernt von dem Pixelschwerpunkt befindet. Dieser Punkt, welcher hier rot markiert ist, kann als Dartspitze bestimmt werden, da sich der Pixelschwerpunkt immer näher an der großen Feder als an der kleinen Spitze befindet.

Abbildung 7 – Erkannte Dartspitze *(eigene Abbildung)*

Wie Abbildung 7 – Erkannte Dartspitze zeigt, kann dieser Algorithmus dann in der Praxis verwendet werden um sehr performant die Dartspitze, und somit den Verbindungspunkt mit dem Dartboard ausfindig zu machen. Um ein möglichst gutes Ergebnis zu erhalten, sollte die Kamera für diesen Schritt etwa in einem 45° Winkel zu dem Board ausgerichtet sein, um den Dartpfeil und auch den Verbindungspunkt möglichst eindeutig zu ermitteln. Auch verhindert dies, dass die Feder des Dartpfeils die Dartspitze bedeckt. In einem solchen Szenario würde der beschriebene Algorithmus zum ausfindig machen der Dartspitze nicht mehr das gewünschte Ergebnis liefern.

3.4 Transformation Dartspitze zum Board

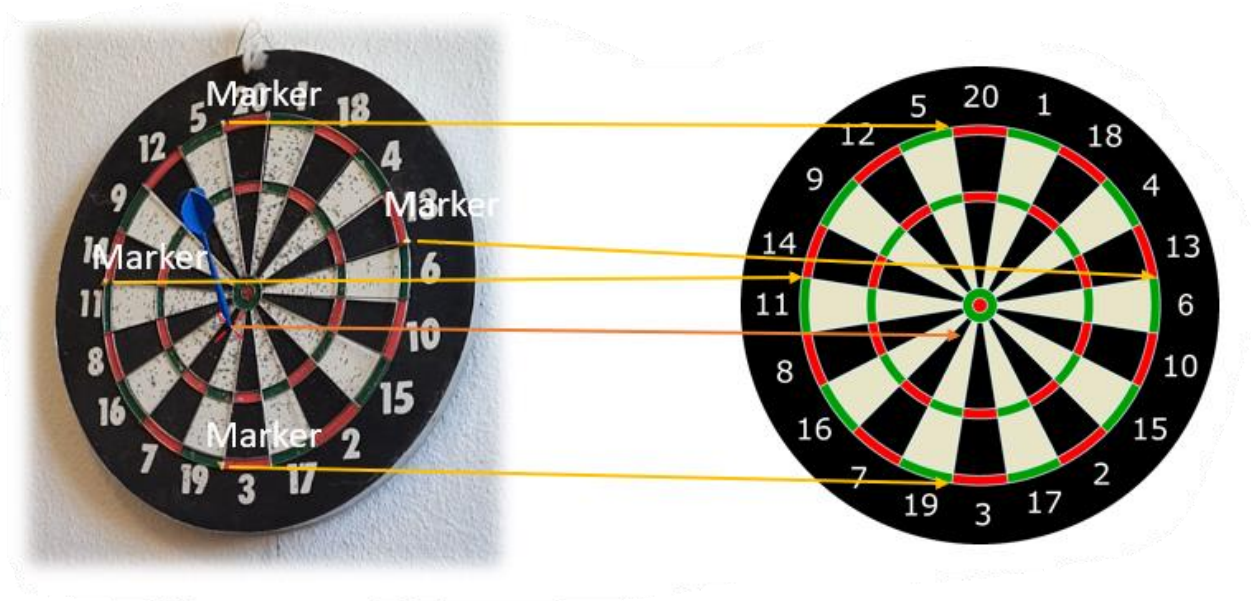

Abbildung 8 – Transformation Dartspitze zum Board (eigene Abbildung)

Da nun bekannt ist, wo sich das Dartboard befindet, wie dieses ausgerichtet ist und wie die Koordinaten der Dartspitze sind, kann die relative Position der Dartspitze zu dem Dartboard bestimmt werden. Dafür werden zunächst die Referenzpunkte aus Kapitel 3.1 verwendet, um eine Transformationsmatrix zu generieren. Diese sagt aus wie die Dartscheibe rotiert, verschoben und verzerrt werden müsste, um zu der imaginären Dartscheide zu kommen, welche auf der

Abbildung 8 – Transformation Dartspitze zum Board auf der rechten Seite abgebildet wird. Mit der erhaltenen Transformationsmatrix kann dann bestimmt werden wo sich die Dartspitze auf diesem imaginären Dartboard befindet, womit man die relative Position zu einer geradlinig ausgerichtet Dartscheide ermittelt hat.

3.5 Spielpunkte ermitteln

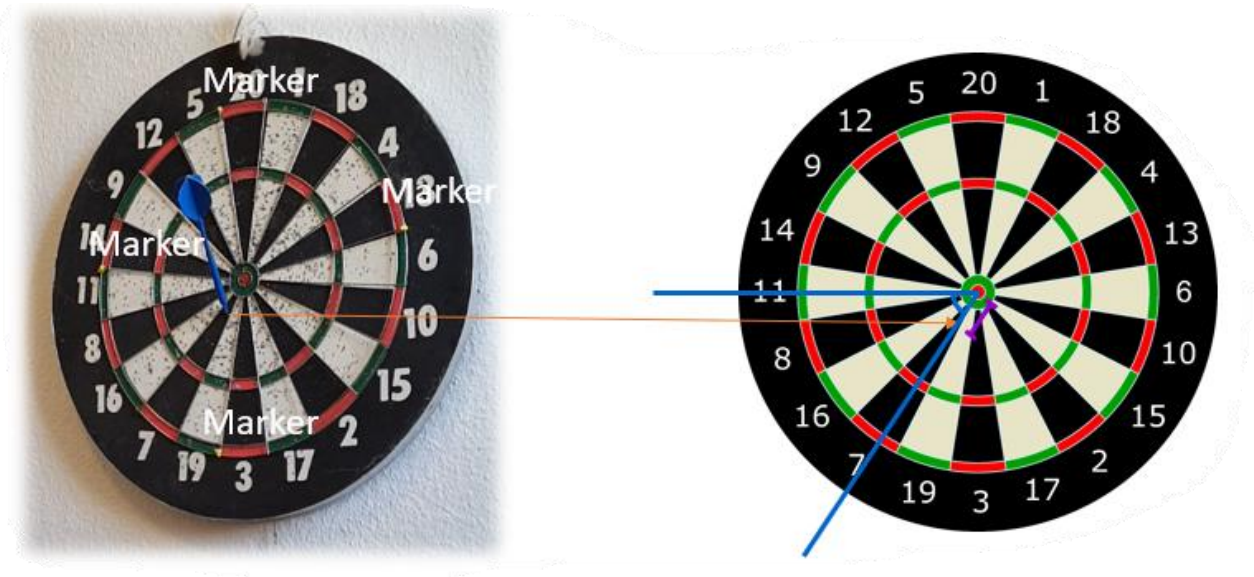

Abbildung 9 – Spielpunkte Ermitteln *(eigene Abbildung)*

Nun gilt es nur noch der Ermittelten Dartspitze einen Wertzuzuordnen. Hierfür wird zunächst der Winkel bestimmt, indem die Dartspitze im Vergleich zur Dartmitte befindet. Da den unterschiedlichen Punktkästchen allen eine gewisse Winkelspanne zugewiesen werden kann, wird in der Abbildung 9 – Spielpunkte Ermitteln bestimmt, dass sich die Dartspitze in der Winkelspanne der 7er Punkte befindet. Nun kann zudem noch über den Abstand bestimmt werden ob nicht sogar doppelte bzw. dreifache Punkte oder das Bull bzw. Bullseye getroffen wurden. Auch ein Fehlwurf, welcher das Board getroffen hat, aber keine Punkte mehr erzielen soll, kann auf diese Weise identifiziert werden. Entsprechend kann über diese beiden Informationen immer eindeutig bestimmt werden welche Punktzahl geworfen wurde, solange das Dartboard überhaupt getroffen wurde.

4 Projektstruktur

Das Gesamtprojekt besteht aus mehreren Teilen, wobei nur „Dartstracker“ für die eigentliche Technische Lösung verantwortlich ist. Im Folgenden wird die grobe Struktur und deren Aufgabenbereich erläutert.

4.1 Java poc

Das Java Maven Projekt „poc“ befindet sich direkt im Wurzelverzeichnis des Gesamtprojekts dem identischen Namen.

/poc

Es folgt der Standard Maven Struktur:

Pfad	Beschreibung / Inhalt
pom.xml	Build-Vorschrift im Maven Stil
/src/main/java	Enthält einige einzeln startbare Java Programme, die bestimmte Teile der Gesamtlösung abbilden
/src/main/resources	Enthält die Ressourcen (Bilder und Videos) für die Java Programme, aber auch die native OpenCV Bibliothek

Tabelle 1 – Maven Struktur (eigene Tabelle)

4.2 Python Skripte

Die Python Skripte befinden unter folgendem Pfad:

/dartstracker/src/test/resources/scripts

Auch diese nutzen Ressourcen, welche sich in folgenden Pfaden finden lassen:

Pfad	Beschreibung / Inhalt
dartstracker/src/test/resources/testInput	Enthält Bilder und Videos die in den Skripten verarbeitet werden
dartstracker/src/test/resources/testOutput	Ausgaben der Skripte werden in diesem Ordner gespeichert

Tabelle 2 – Python Struktur (eigene Tabelle)

4.3 Dokumentation

Die Projektberichte und Präsentationen aus dem ersten und zweiten Semester befinden sich in dem Ordner:

/Dokumentation

4.4 Dartstracker

Die gesamte Applikation „dartstracker“ befindet sich im Wurzelverzeichnis des Projekts. Das Projekt orientiert sich dabei an die für Gradle-Buildsysteme typische Struktur:

Pfad	Beschreibung / Inhalt
build.gradle gradle.properties settings.gradle	Globale Gradle „Build“-Anleitung und Einstellungen

Tabelle 3 – Globale Gradle Build Dateien (eigene Tabelle)

Dabei gliedert sich das Projekt in 2 Module auf:

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

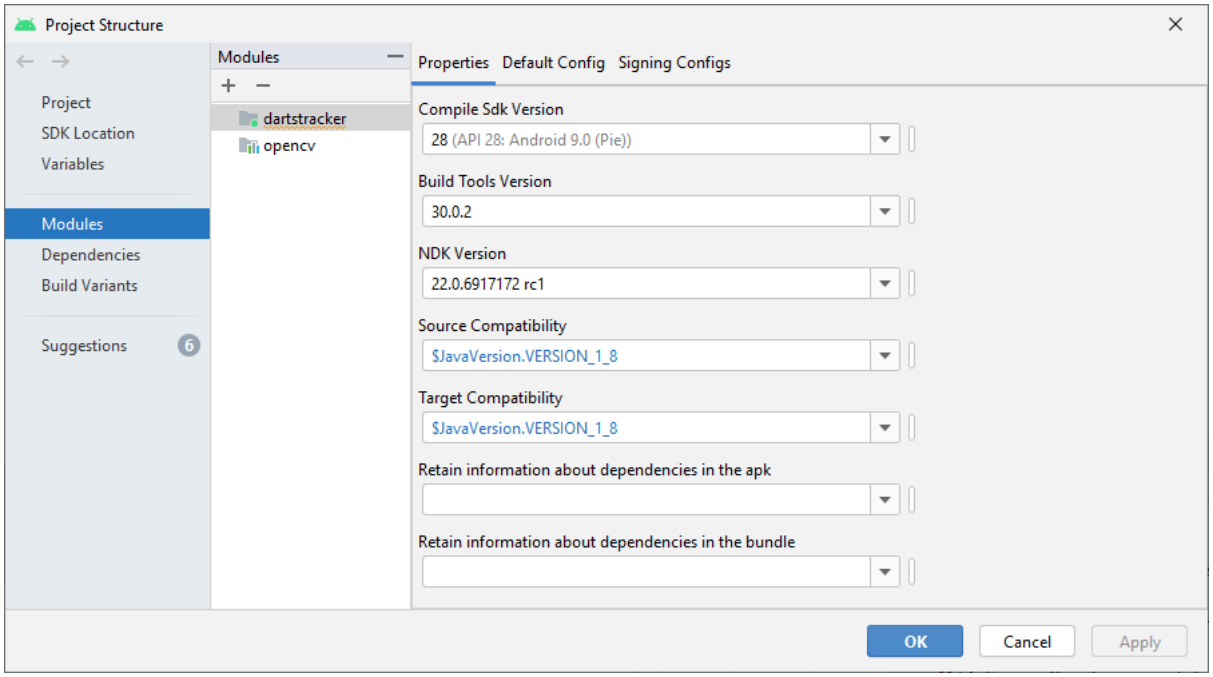

Abbildung 10 – Projektmodule (eigene Abbildung)

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

4.4.1 Modul opencv

Das opencv modul ist ein Import der original Android SDK. Dafür muss der Pfad zuvor in der übergeordneten „settings.gradle“ eingetragen werden. Die Struktur wird unverändert beibehalten wie es die SDK vorgibt.

4.4.2 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 einzelnen Funktionen der verwendeten Bilderkennung widerspiegeln ohne jedoch ein gesamtes Android Betriebssystem hochfahren zu müssen. Weitergeführt wurden die Tests jedoch in dem separaten Projekt „poc“ bzw. in den Python Skripten.
dartstracker\build.gradle	Wichtigste „Build“- Anleitung für den DartsTracker. Enthält Beschreibung sämtlicher Abhängigkeiten sowie externe Builds

Tabelle 4 – „dartstracker“ Struktur (eigene Tabelle)

Für die Oberflächendesigns existieren eine Reihe von Design-XMLs in folgenden Ordnern:

Pfad	Beschreibung / Inhalt
res/drawable	Icons
res/layout	Activity Designs
res/menu	Menu Designs
res/mipmap	App Icons für das Android Betriebssystem
res/values	Texte und Styles

Tabelle 5 – „dartstracker“ Ressourcen (eigene Tabelle)

Um die benötigten Funktionen zu kapseln, wurde eine Reihe von Java sowie einer C++ Pakete und Klasse erstellt. Die folgende Übersicht gibt einen groben Überblick über die generellen Aufgaben der Pakete deren Zusammenhänge. Hierbei wird kein Anspruch auf Vollständigkeit erhoben. Die Übersichten dienen lediglich als Einstiegshilfe um sich zügig im Projekt zurecht zu finden.

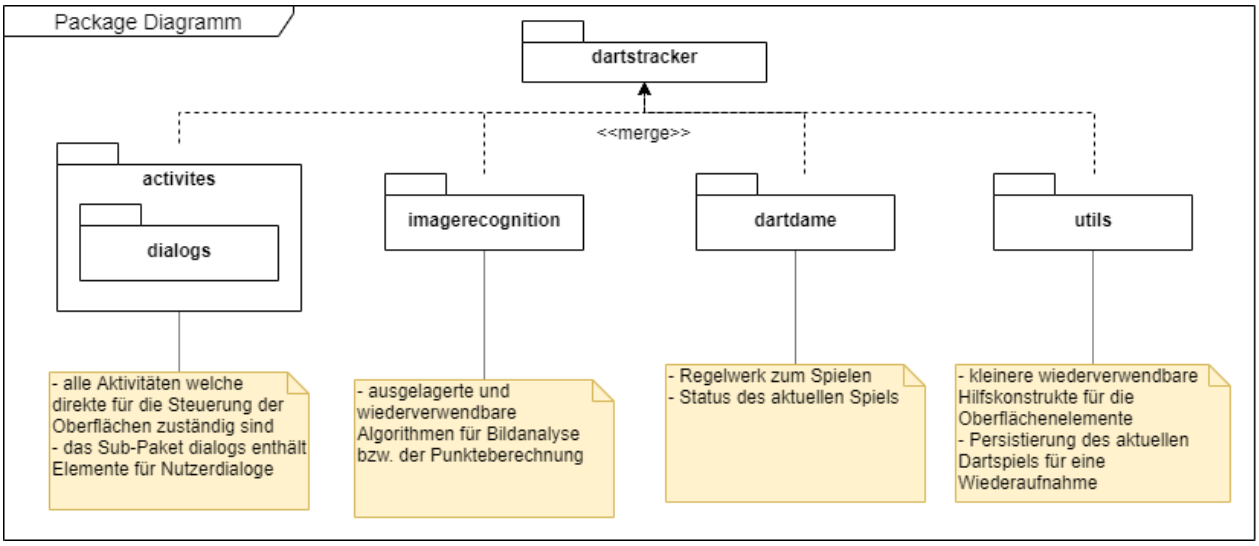

Abbildung 11 – Package Diagram „dartstracker“ (eigene Abbildung)

Das Activities-Paket enthält die zentralen Java-Klassen, welche in Android die Steuerung der Oberflächen übernehmen. In dem Paket befinden sich auch die Dialog-Klassen die auf den Oberflächen Verwendung finden. Die folgende Übersicht verdeutlich den inhaltlichen Zusammenhang der Klassen.

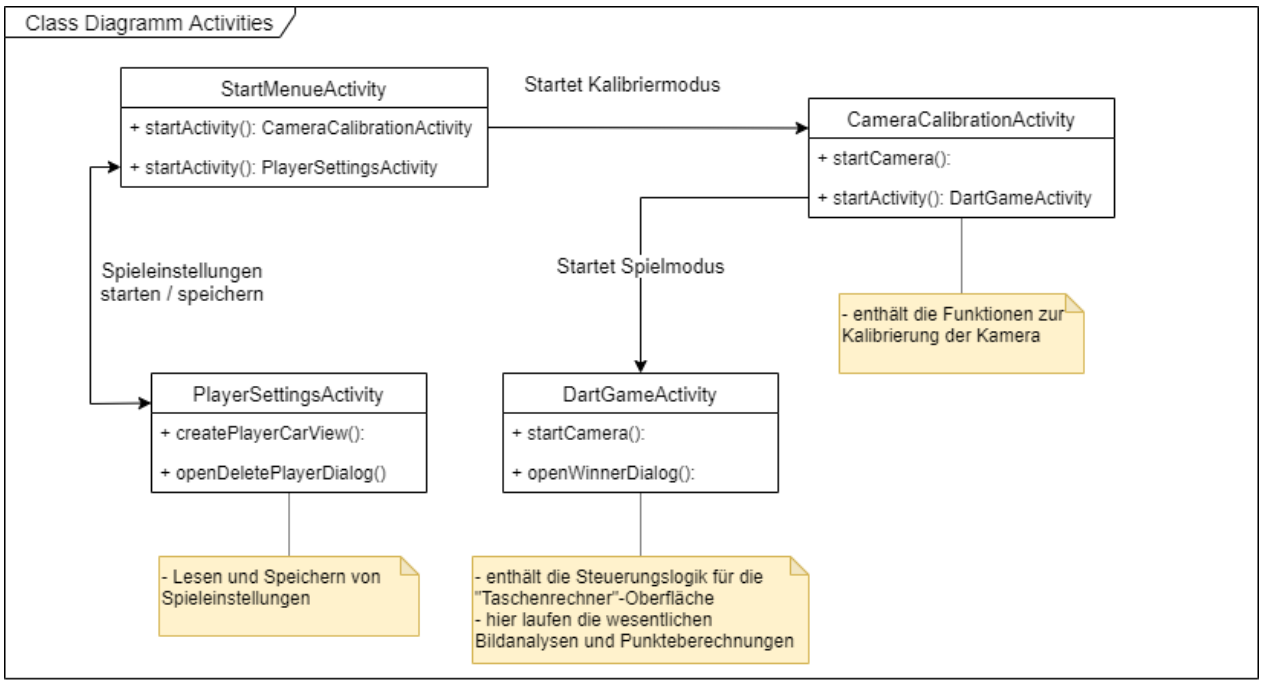

Abbildung 12 – Class Diagramm „Activities“ (eigene Abbildung)

5 Benutzeroberfläche

Das Frontend der *Darts Tracker* App kann in vier funktionale Bereiche unterteilt werden. Zu den Bereichen gehört der Startbildschirm, die Spieleinstellungen, die Kalibrierung und die Spieloberfläche in Form eines Taschenrechners.

5.1 Startbildschirm

Das Startmenü der *Darts Tracker* App fungiert als Startpunkt der Nutzung der App. Die Oberfläche ist unterteil in zwei Oberflächenelemente vom Typ „CardViews“, die die Navigation in die Bereiche „Spiel Starten“ und „Spieleinstellungen“ ermöglicht.

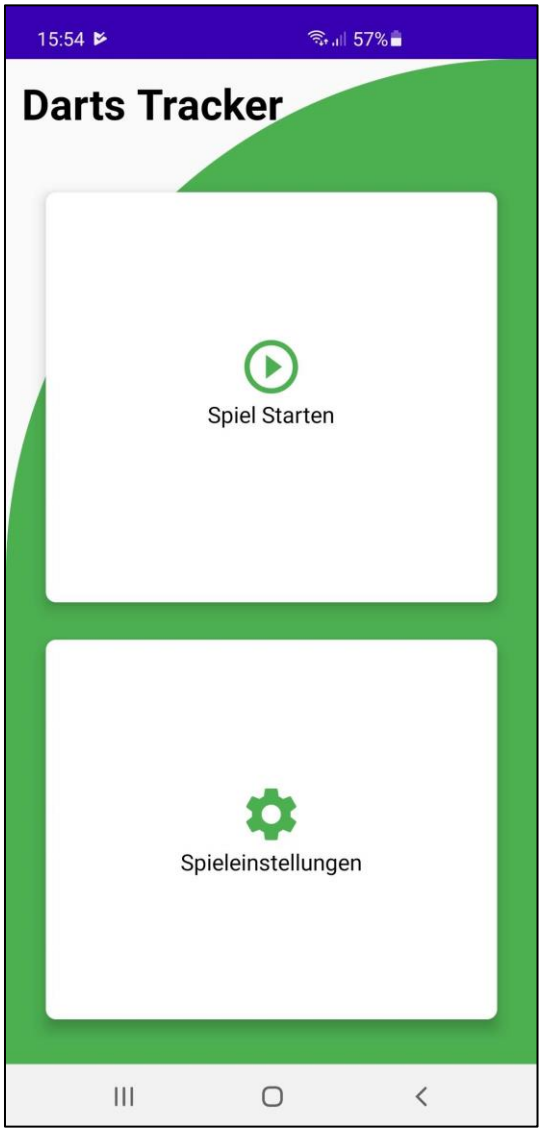

Abbildung 13 – Startmenü (eigene Abbildung)

Der „Spiel Starten“-Oberflächenbaustein verfügt über die Möglichkeit den Nutzer in die Kalibrierungsphase zu navigieren. Die Navigation ist beim erstmaligen Starten deaktiviert,

da der Nutzer zunächst Spieleinstellungen vornehmen muss, bevor das Spiel gestartet werden kann.

Über die „Spieleinstellungen“ kann der Nutzer in die gleichnamigen Einstellungen wechseln.

5.2 Spielereinstellungen

Die Spieleinstellungen erfüllen die Funktionalität, Voreinstellungen für das Dart-Spiel vorzunehmen. Die Oberfläche besteht aus drei „CardViews“. Darin hat der Nutzer die Möglichkeit neue Spieler hinzuzufügen, Spieler zu löschen und den Spielmodus auszuwählen (501 oder 301).

Abbildung 14 – Spieleinstellungen *(eigene Abbildung)*

Die Funktionalität „Player hinzufügen“ sorgt bei der Betätigung der „Card View“ dafür, dass ein Dialog-Fenster geöffnet wird, in dem der Nutzer einen neuen Spieler anlegen kann. Dies geschieht über die Eingabe des Namens und dem Bestätigen des Oberflächenbausteins „Spielernamen Bestätigen“.

Abbildung 15 – Player hinzufügen Dialog *(eigene Abbildung)*

Im Anschluss an die Bestätigung wird der neue Spieler über ein Textfeld innerhalb der „CardView“ angezeigt.

Die Funktionalität „Player Löschen“ funktioniert nach einem ähnlichen Verfahren. Sofern sich der Nutzer entschließt einen Spieler löschen zu wollen, öffnet sich ein Dialog – Fenster, in dem die bereits erstellten Spieler angezeigt werden. Mit einem Klick auf den jeweiligen Namen oder mit dessen manuellen Eingabe, kann die Löschung bestätigt werden und der Spieler wird entfernt.

Abbildung 16 – Player löschen Dialog *(eigene Abbildung)*

5.3 Kalibrierung

Die Kalibrierungsansicht besteht im Wesentlichen aus 4 Teilen, einem „ImageView“ für die Bildvorschau, einer „ProgressBar“ für die Erkennungsqualität der Marker, einer „SeekBar“ für die manuelle Einstellung des Kamera Zooms und einem Knopf zum Überspringen dieser Ansicht.

Sofern die Kamera alle 4 Marker stabil für einige Sekunden erkannt hat, wird das Spiel automatisch gestartet.

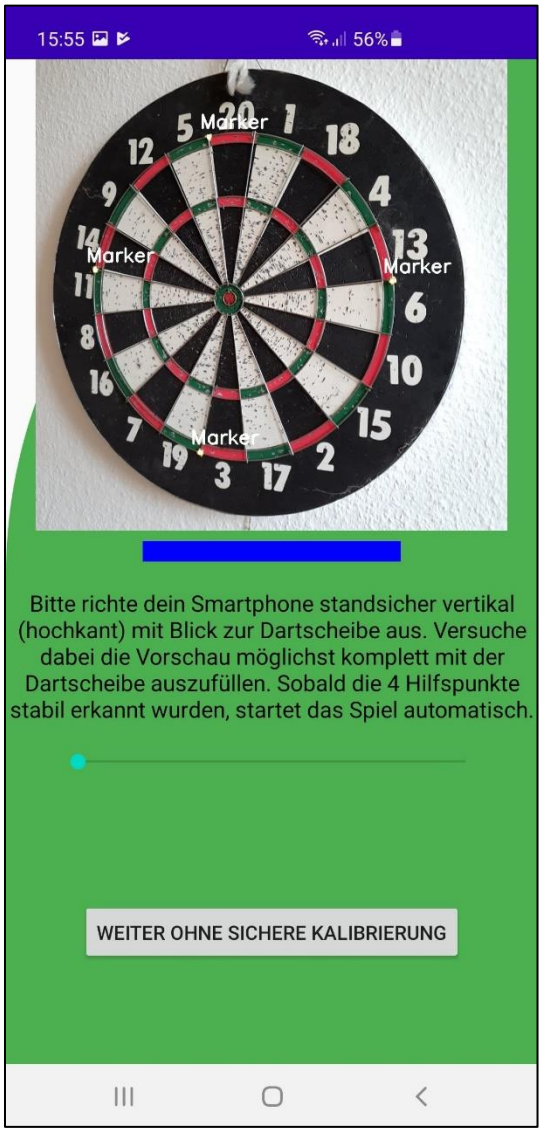

Abbildung 17 – Kalibrierung *(eigene Abbildung)*

5.4 Spielansicht

In der Spielansicht befinden sich eine Vielzahl von „CardViews“ und Textelementen für die Anzeige des „Taschenrechners“. Dieser dient für die manuelle Eingabe oder Korrektur von Pfeilerkennungen. Im oberen Bereich ist der aktuelle Stand des Spiels erkennbar sowie – farblich hinterlegt – welcher Spieler aktuell an der Reihe ist. Mit dem „Verbotszeichen“ in der rechten oberen Ecke kann das Spiel auch unterbrochen werden.

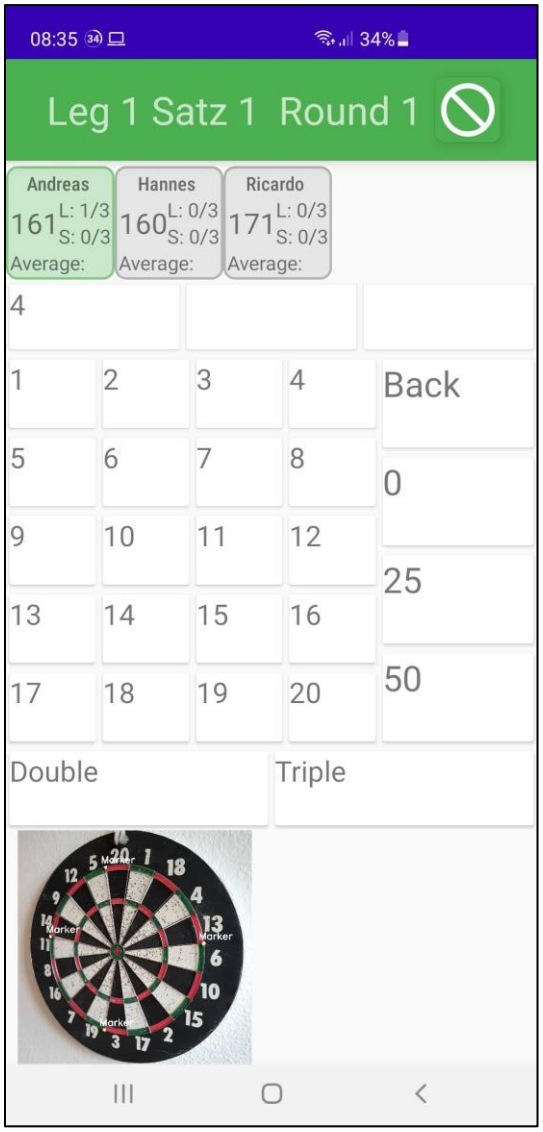

Abbildung 18 – Spielansicht *(eigene Abbildung)*

Im unteren linken Bereich befindet sich noch mal eine Kameravorschau. Auch werden die Marker und ggf. erkannten Pfeile markiert. Sie dient als Kontrolle für die korrekte Ausrichtung des Smartphones. Rechts neben dem Vorschaubild wird dann der Spielwert angezeigt, der aktuell bei einem Pfeil erkannt wurde.

5.5 Gewinneransicht

Sofern das Spiel durch einen Spieler gewonnen wird, also im 301 o. 501 Spielmodus auf 0 heruntergespielt hat, wird automatisch der Gewinnerbildschirm angezeigt mit dem entsprechenden Sieger. Dieses kann mit einem Klick beendet werden und der Spielstatus wird in der Spielansicht aktualisiert und eine neue Runde gestartet.

Abbildung 19 – Gewinneransicht (eigene Abbildung)

6 Performancemessung

Um die Leistungsfähigkeit der technischen Lösung zu überprüfen, wurde eine kurze Messreihe durchgeführt. Diese folgt keinen strengen wissenschaftlichen Standards und erhebt damit auch nicht den Anspruch einer zweifelsfreien exakt reproduzierbaren Messung, sondern stellt lediglich eine Stichprobe dar mit Materialien, die dem Team zur Verfügung standen. Diese Performancemessung kann als Anhaltspunkt zum Aufbau für den Betrieb der Darts Tracker App dienen oder als Vergleichswert sofern sich technische Lösung oder Rahmenbedingungen ändern.

Rahmenbedingungen:

Um dennoch die Messung möglichst nachvollziehbar zu halten, werden im Folgenden die Rahmenbedingungen beschrieben unter welchem die Messung stattgefunden hat.

- Test-Smartphone Modell war ein Galaxy S10+
- Smartphone wurde auf einem festem Kamerastativ angebracht mit Blickrichtung auf das Board-Zentrum (siehe Abbildung 20 – Schematischer)
- Kalibrierung ohne Zoom Nutzung – alle 4 Marker wurden einwandfrei erkannt. Dadurch befindet sich das Testgerät relativ dicht am Board, behindert jedoch nicht die Wurfzone
- Handelsübliches Dartboard frei hängend auf einer weißen Wand angebracht
- Es wurde Tageslicht, jedoch ohne direkt Sonneneinstrahlung genutzt (zum Zeitpunkt der Messung war das Wetter bedeckt)
- Die Bildrate wurde auf ~15 Bildern/s eingestellt
- Jedes Bild hatte eine Auflösung von 1280 x 960 Pixeln

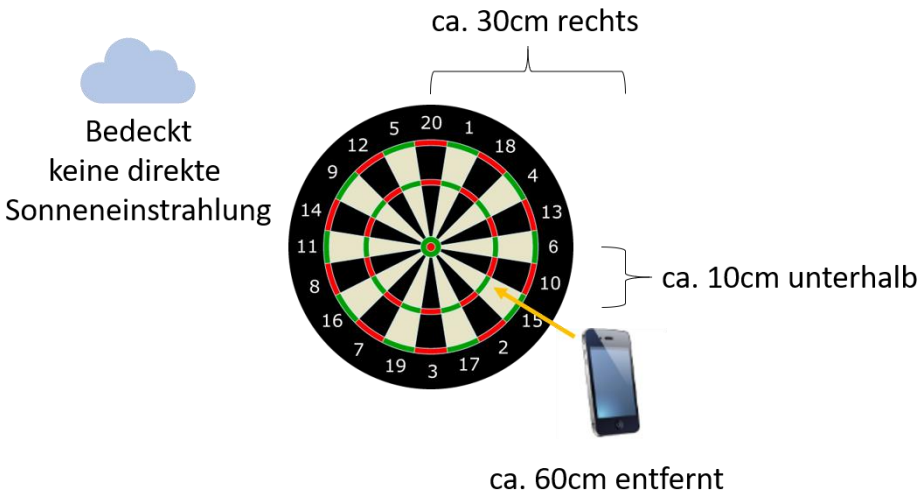

Abbildung 20 – Schematischer Testaufbau (eigene Abbildung)

Messung:

Für die Messung wurde jeweils ein blauer Pfeil geworfen und nach einem erfolgreichen Wurf wieder vom Board entfernt. Somit steckte in dem Board immer nur maximal ein Pfeil gleichzeitig. Würfe wurden nur gezählt, wenn diese auch tatsächlich im Board stecken blieben – sei es in einer gewerteten Zone oder im Aus. Alle anderen Würfe gingen nicht in die Wertung ein. Insgesamt wurden 30 Treffer gewertet.

Es wurden 3 Ergebniskategorien festgelegt:

- **Korrekt erkannt (grün)** – Vom Bildanalysesystem wurde der Pfeil korrekt erkannt und die Punkte dazu ermittelt
- **Schwankend erkannt (gelb)** – Vom Bildanalysesystem wurde der Pfeil zwar erkannt aber die Punkteermittelung ist nicht stabil und schwankt zwischen verschiedenen Punktezonen. Pendelt sich oft nach kurzer Zeit auf einen Wert ein.
- **Falsch erkannt (rot)** – Entweder wurde der Pfeil falsch oder gar nicht erkannt oder die Punkte falsch ermittelt.

Kategorie	Anzahl Treffer
Grün	25
Gelb	2
Rot	3
Gesamt	**30**

Tabelle 6 – Performancemessung (eigene Tabelle)

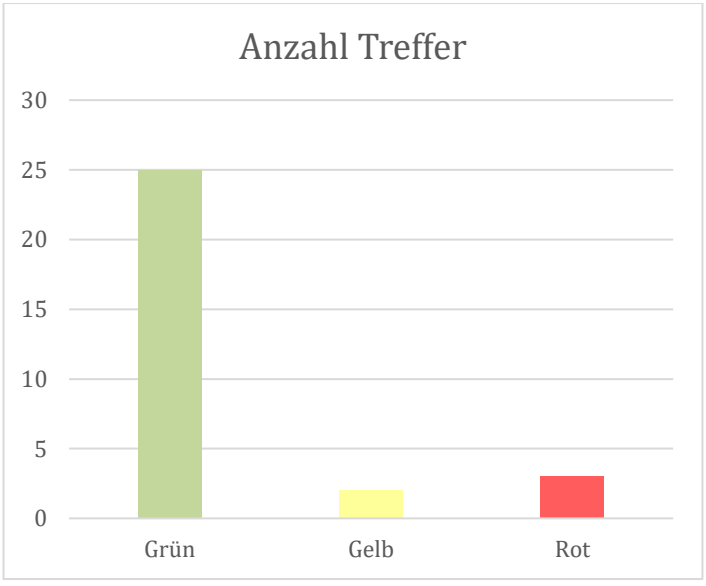

Abbildung 21 - Balkendiagramm zur Performancemessung (eigene Abbildung)

7 Auswertung

7.1 Diskussion der Ergebnisse

Im Rahmen des Teamprojekts konnte das Team zeigen, dass es technisch möglich ist, Treffer im Spiel Darts automatisiert und in Echtzeit zu analysieren, zu bewerten und diese in eine automatische Zählung in dem Spiel zu integrieren. Darüber hinaus gelang dies auf handelsüblichen Smartphones wie sie heutzutage die meisten Menschen bei sich tragen. Dies ermöglichte auch eine handhabbare Spieloberfläche inklusive einer manuellen Eingabe von Dartwürfen zu programmieren.

Die einfache Performancemessung zeigt, dass die technische Lösung bereits für das Team auf einem akzeptablen Niveau liegt. 83% der Würfe wurden dabei sicher erkannt. Die 2 schwankenden Würfe müssen schlussendlich nicht falsch im Spiel gewertet worden sein, werden dennoch als Nicht-Erreichen des Ziels (einer sicheren Erkennung) gewertet. Damit verbleibt die Leistung bei 25 von 30 richtig erkannten Würfen.

Während der Erarbeitung dieser technischen Lösung konnten zahlreiche Ursache für Ungenauigkeiten und Fehlerquellen ausgemacht werden. Auch wenn diese nicht durch empirische Messungen des Teams bewiesen wurden, können sie doch als verlässliche Hinweise genutzt werden.

- Einige Teilschritte der Lösung basieren vollständig auf Farbfilterung. Die Konvention für einfarbige blaue Pfeile und den 4 gelben Markern ist sogar regelkonform, da es schlicht keine Vorgaben dazu gibt. Jedoch ist die Erkennungsqualität stets abhängig von den genutzten Spielmaterialien (leicht unterschiedliche Farben bei Board, Pfeil oder Marker), der genutzten Hardware (Farben nimmt jedes Smartphone leicht unterschiedlich auf) und letztlich auch vom einfallenden Licht und dem Hintergrund.

- Auch wenn man es mit bloßem Auge nicht sehen kann, treten bei jedem Wurf, bei jeder Bewegung im Raum Mikrobewegungen an der Dartscheibe oder bei dem verwendeten Smartphone auf. Dieses Problem konnte jedoch nahezu vollständig gelöst werden durch die Beschleunigung des Analysealgorithmus, so dass bei jedem einzelnen Frame prinzipiell ein unabhängiges Ergebnis bereitgestellt werden kann. Das Problem tritt jedoch sofort wieder auf, sofern Analyseergebnisse aus vorherigen Frames verwendet werden, wie es z. B. bei Caching Mechanismen vorkommen könnte.

- Die technische Lösung betrachtet stets nur einen Pfeil gleichzeitig. Nach offiziellen Regeln müssten jedoch bis zu 3 Pfeile bei einem Spieler stecken bleiben. Damit gehen sofort Überdeckungsproblematiken einher die in dieser Lösung explizit nicht betrachtet wurden.

- Ungenauigkeiten gibt es bereits bei der exakten Bestimmung der Marker auf dem Board (Erkennung wird gemittelt) und durch die mögliche Überdeckung durch den Draht auf dem Board, sofern ein Pfeil sehr dicht an einem solchen einschlägt. Diese Ungenauigkeiten bewegen sich jedoch bereits im niedrigen einstelligen mm-Bereich.

7.2 Ausblick

Neben den noch vorhandenen Einschränkungen und Ungenauigkeiten wurden auch eine Vielzahl an Verbesserungsmöglichkeiten gesammelt. Die könnten als Ideen aufgegriffen werden für eine Fortführung des Projekts.

7.2.1 Borderkennung

Über andersfarbige Referenzpunkte: Die derzeitig verwendeten gelben Referenzpunkte sind einfach gestrickt, weshalb sie auch entsprechend fehlerfrei bei korrekter Anwendung sind. Nichtsdestotrotz hat man nicht immer perfekt abgestimmte Gelbe Punkte zur Hand. Hierfür würde es sich anbieten als ersten Kalibrierungsschritt ein Bild von einem Referenzpunkt zu machen, damit Farbe und ggf. Große in Abhängigkeit der gegebenen Umstände automatisiert optimal abzustimmen.

Ohne Referenzpunkte: Wie bereits in den vorherigen Kapiteln erläutert, werden 4 gelbe Referenzpunkte benötigt, um die Erkennung des Dart Boards zu gewährleisten. Diese Notwendigkeit könnte durch die Nutzung eines SIFT Algorithmus inklusive eines geeigneten Referenzbildes abgelöst werden. Dabei besteht die Chance die grundsätzliche Genauigkeit zu steigern und die kritischen Umwelteinflüsse wie Lichtfarbe und -intensität zu verringern.

7.2.2 Pfeilerkennung ohne einfarbige Pfeile

Über Differenzbildung: Ein zwischenzeitlich verfolgter Ansatz bei der Pfeilerkennung war es ein Differenzbild zwischen aufeinander folgenden Frames zu ermitteln. Hierbei werden die Pixelwerte miteinander abgeglichen und die Unterschiede hervorgehoben. Von einem Dartpfeil konnte dann ausgegangen werden, wenn ein hinzugekommenes Objekt mit in etwa erwarteter Größe stabil über mehrere Frames aufgenommen wird. Leider ist diese Methode recht fehleranfällig gegenüber Verwacklungen und anderen Störfaktoren. Entsprechend mussten einige Bildoptimierungen durchlaufen werden, um das Rauschen zu minimieren ohne Datenverlust, wie eine Abgerundete Dartspitze, zu erhalten. Auch die Dartspitzenerkennung aus Kapitel **Fehler! Verweisquelle konnte nicht gefunden w erden.** musste überarbeitet werden, um mehrere Große Pixelmassen entgegenzunehmen, da Abschnitte des Pfeils ggf. zu identisch mit dem dahinter liegenden Dartboard waren um als Teil eines neuen Objektes erkannt zu werden. Schlussendlich wurde ein vergleichbar gutes Ergebnis erreicht, welches jedoch nicht ansatzweise so effektiv wie die Verwendung der Farbigen Pfeile war. Nichtsdestotrotz liegen noch immer Unit Tests vor, um zu einem Späteren abschnitt Abschnitte wiederzuverwenden.

Mit KI-Unterstützung: Pfeile müssen in der aktuellen Lösung einfarbig (blau) sein. Typischerweise sind Pfeile jedoch bunt bzw. metallisch Farben. Die Pfeile vom Board zu unterscheiden ist typische *Background Subtraction* Aufgabe. OpenCV bietet sogar eine ganze Reihe von verschiedenen Algorithmen diesbezüglich an. Jedoch waren die Ergebnisse nicht befriedigend, weshalb dieser Lösungsweg zunächst nicht weiterverfolgt wurde. Der Arbeitsstand hierzu befindet sich noch in den Python Skripten. Sofern hierbei bessere Ergebnisse erzielt werden, ist es vorstellbar, dass Pfeile in beliebiger Größe, Farbe und

sogar Form genutzt werden könnten. Auch wäre damit eine Möglichkeit eröffnet mehrere Pfeile von unterschiedlichen Würfen auf dem Board zu unterscheiden.

7.2.3 Koppelung von Kameras (Blickwinkel)

Die aktuelle Lösung kann nur eine Kamera eines Smartphones nutzen. Dies hat stets den Nachteil, dass es nur einen Blickwinkel gibt und spätestens bei mehreren Pfeilen das Problem der Überdeckung kritisch wird. Es wäre denkbar mehrere Smartphones zu koppeln z. B. per Bluetooth, um so verschiedene Blickwinkel gleichzeitig auswerten zu können. Im Besten Falle eliminiert man damit das Problem der Überdeckung komplett.

7.2.4 Konfiguration aller Schwellwerte

Aktuell gibt es eine ganze Reihe an Schwellwerten innerhalb der Applikation. Diese werden benötigt, um Farbbereiche einzugrenzen z. B., um festzulegen was als gelb oder blau definiert wird. Aber auch Leistungswerte sind innerhalb der Applikation fest definiert. So z. B. die aufgenommenen Pixel pro Bild oder generell die Anzahl der Bilder pro Sekunde. Auch müssen bestimmte Mengen an Werten erst überschritten werden um nicht als Rauschen definiert zu werden. Während der Bearbeitung wurde bereits festgestellt, dass einige Werte Modellabhängig vom verwendeten Smartphone sind. Deutlich wurde das bei der Definition Farben. Die Unterschiede sind zwar nicht groß, können aber dennoch zur signifikanten Verschlechterung der Erkennung führen, weshalb alle diese Schwellwerte prinzipiell über den Kalibiermodus ggf. frei einstellbar sein sollten. Damit könnte man die Lauffähigkeit auf vielen unterschiedlichen Geräten gewährleisten und ggf. sogar den Akku schonen.

8 Abbildungsverzeichnis

9 Tabellenverzeichnis