UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Bildbasierte Integration von Software am Beispiel der Entwicklung eines Expertensystems für Online-Poker-Plattformen

BACHELORARBEIT

zur Erlangung des Grades eines
Bachelor of Science

im Studiengang Computervisualistik vorgelegt von
Olaf Poneta

und im Studiengang Informatik vorgelegt von
Jan-Hendrik Borth

Erstgutachter
Prof. Dr. Ralf Lämmel

Zweitgutachter
Dr. Markus Kaiser

Koblenz, im Dezember 2011

Erklärung

Ich versichere, dass ich meinen gekennzeichneten Teil, der vorliegenden Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☐ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☐ nein ☐

Koblenz, den 12.12.2011

Inhaltsverzeichnis

Abstract **iv**

1 Einleitung **1**
1.1 Motivation 1
1.2 Aufbau 2

2 Integration von Software **4**
2.1 Definition Softwareintegration 4
2.2 Einsatz von Softwareintegration 4
2.3 Interaktion mit vorhandener Software 5
2.3.1 Softwareintegration mit einem Wrapper 5
2.3.2 Integration über Hardwareschnittstellen 5
2.3.3 DLL-Injection, Binary Patching und Runtime Patching 6
2.4 Reverse Engineering 6
2.4.1 Interne Informationsextraktion 7
2.4.2 Reverse Engineering ohne Quellcode 7
2.5 Bildbasierte Integration von Software 8
2.5.1 Screen Scraping 9
2.5.2 Stand der Technik beim Screen Scraping 10
2.5.3 Vorzüge der bildbasierten Softwareintegration 10
2.5.4 Nachteile der bildbasierten Softwareintegration 10

3 Einführung in die Domäne Poker **12**
3.1 Das Pokerspiel 12
3.1.1 Spielvarianten 13
3.1.2 Regeln 13
3.1.3 Pokerstrategie 15
3.2 Anleitung nach Pokerstrategy.com 20
3.2.1 Das Spiel vor dem Flop 20
3.2.2 Das Spiel nach dem Flop 21
3.2.3 Diskussion der Anleitung 22
3.3 Online-Poker 23
3.3.1 Ablauf eines Onlinespiels 23

3.3.2 Grafische Darstellung von Pokerplattformen 24
3.4 Pokersoftware 25
3.4.1 Wissenschaftliche Pokerbots 26
3.4.2 Pokerbots für Online-Poker-Plattformen 26

4 Konzept der entwickelten Software 29
4.1 Anforderungen an den PokerBot 29
4.1.1 Spielerische Anforderungen 30
4.1.2 Technische Anforderungen 30
4.2 Module 30
4.2.1 Modul 1: Informationsgewinnung 31
4.2.2 Modell des aktuellen Spielzustands 32
4.2.3 Modul 2: Spiellogik 32
4.2.4 Modul 3: Interaktion 36
4.3 Weitere Bestandteile von PokerBot 36
4.3.1 Grafische Benutzeroberfläche 36
4.3.2 Test- und Analysewerkzeuge 38

5 Implementierung 41
5.1 Modulübergreifende Programmteile 41
5.1.1 Unterstützung verschiedener Pokerplattformen 41
5.1.2 Datenstrukturen 43
5.1.3 Initialisierungsroutine 45
5.1.4 Hauptschleife 46
5.2 Modul 1: Informationsgewinnung 48
5.2.1 Template Matching 48
5.2.2 Texterkennung 50
5.2.3 Fenstererkennung 53
5.2.4 Aktualisierung des Spielzustandsmodells 54
5.3 Modul 2: Spiellogik 58
5.3.1 Interpretation des Modells 58
5.3.2 Formalisierung des Regelsystems 61
5.3.3 Stochastische Überprüfung 63
5.3.4 Nachkontrolle 66
5.4 Modul 3: Interaktion 67

6 Auswertung 69
6.1 Aufwand der bildbasierten Softwareintegration 69
6.2 Performance der Softwareintegration 69
6.3 Qualität der bildbasierten Softwareintegration 69
6.3.1 Fehler beim Template Matching 70
6.3.2 Fehler bei der Interpretation 70
6.3.3 Fehler bei der Interaktion 71
6.4 Spielerische Qualität von PokerBot 71

6.5 Ausblick 72

Abstract

Software integration is an engineering task where image-based approaches are still considered slow and error-prone. In this thesis, we apply image-based software integration to the domain of online Poker systems. That is, we implement the poker expert system PokerBot which can play online. We use the method of screen scraping to capture the screen information needed to interact with the Poker server. The consequent use of Template Matching leads to an efficient implementation. Substantial effort was also addressed to the artificial intelligence aspects of PokerBot. The purpose of AI is here to mimic human playing style by translating a well-documented Pokerguide into formalized, executable language.

Kapitel 1

Einleitung

Der Großteil aller Softwareprodukte ist darauf ausgelegt, mit dem Endbenutzer zu interagieren. Eine übersichtliche und durchdachte Darstellung, sowie eine intuitive und ergonomische Steuerung, sind wesentliche Bestandteile aktueller Softwareprodukte. Soll ein Programm mit einem anderen Programm interagieren, um komplexere oder interdisziplinäre Aufgaben zu lösen, geschieht das über speziell dafür vorgesehene Schnittstellen. Besonders bei interdisziplinären Aufgabenstellungen kann es vorkommen, dass notwendige Schnittstellen nicht vorhanden sind. Diese müssen dann nachträglich implementiert werden oder die Zusammenarbeit mehrerer Programme kann nicht automatisiert stattfinden. In so einem Fall muss der Benutzer die Aufgabe mit hohem Aufwand und der iterativen Zuhilfenahme aller benötigten Programme lösen.

Häufig ist das nachträgliche Implementieren von Schnittstellen oder die klassische Integration von mehreren Anwendungen, unverhältnismäßig aufwändig. Maßgebend für den hohen Aufwand, können etwa fehlender Zugriff auf den Quellcode oder eine unzureichende Dokumentation der Software sein. Auch die Größe und Unübersichtlichkeit vieler Softwareprojekte und die damit verbundene hohe Einarbeitungszeit, sind gewichtige Hindernisse bei der Integration. Von dem hohen Aufwand unabhängig, ist die nachträgliche Änderung von bestehender Software mit Risiken wie der Inkompatibilität bei Aktualisierungen, Versionskonflikten oder sogar Urheberrechtsverletzungen, verbunden.

1.1 Motivation

Computerprogramme und Roboter sind bereits in der Lage Gesichter auf Fotos zu identifizieren [33], Automobile im Straßenverkehr zu steuern [11] und unzählige andere, hochkomplexe Aufgaben zu Lösen. Solche Systeme werden als Expertensysteme [17] bezeichnet. Ein Expertensystem zu entwickeln, welches bestehende Softwareprodukte so benutzt wie es ein mensch-

licher Anwender tut, erscheint gegenüber dem aktuellen Stand der Technik als überschaubare Aufgabe. Die Anforderungen, um ein solches System zu entwickeln, sind minimal. Lediglich die Kenntnis der Benutzeroberfläche, der zu integrierenden Software, ist dazu notwendig. Die Einarbeitung in bestehende Softwareprojekte, die Abhängigkeit von bestehendem Quellcode und dessen Dokumentation, sowie alle anderen Probleme der klassischen Integration von Software entfallen dagegen.

Dieser alternative Ansatz der Softwareintegration wird in älterer Literatur häufig als Screen Scraping bezeichnet. In aktueller Literatur wird der Begriff Screen Scraping in anderem Zusammenhang verwendet, weshalb in dieser Arbeit die Bezeichnung bildbasierte Softwareintegration benutzt wird. Bildverarbeitung gilt in der Softwareentwicklung immer noch als sehr rechenaufwendig, kompliziert und fehleranfällig, weshalb generell, bei informationstechnischen Aufgabenstellungen wie auch der Softwareintegration, möglichst darauf verzichtet wird. Der technologische Fortschritt und insbesondere die Errungenschaften auf dem Gebiet der Bildverarbeitung lassen aber vermuten, dass es durchaus sinnvolle Einsatzgebiete für die bildbasierte Softwareintegration geben kann.

1.2 Aufbau

Diese Arbeit untersucht an einem realistischen Szenario den Ansatz der bildbasierten Integration von Software hinsichtlich seiner Praktizierbarkeit. Als praktische Aufgabe wird die Entwicklung eines Expertensystems für Online-Poker-Plattformen durchgeführt. Die wichtigsten, zu untersuchenden Aspekte sind der Aufwand der Implementierung, die Performance und die Qualität dieses Ansatzes. Die Arbeit gliedert sich in sechs Kapitel:

- Im ersten Kapitel wird die Idee und der Nutzen bildbasierter Softwareintegration beschrieben und vorgestellt, welche Fragestellungen diese Arbeit motivieren.
- Das zweite Kapitel behandelt das Thema: Integration von Software. Der Fokus des zweiten Kapitels liegt darauf, den bildbasierten Ansatz und die derzeit geläufigen Ansätze gegenüberzustellen.
- Das dritte Kapitel führt in die Domäne Poker ein. Zunächst wird dabei theoretisches Grundwissen zu den Regeln und der Strategie von Poker vermittelt, bevor auf die Abläufe beim Onlinepoker und schließlich auf Poker-Expertensysteme eingegangen wird.
- Im vierten Kapitel geht es um das Konzept der im Rahmen dieser Arbeit entwickelten Software. Dabei werden die einzelnen Bestandteile und die Gesamtarchitektur vorgestellt.
- Die wichtigsten Themen der Implementation werden im fünften Kapitel vertieft. Insbesondere werden die Module, sowie Abläufe und Daten-

strukturen mit vielen Programmbeispielen veranschaulicht.

- Eine Auswertung der gewonnen Erkenntnisse erfolgt im sechsten Kapitel.

Kapitel 2

Integration von Software

Sollen in einer Anwendung Teile aus unterschiedlichen Softwareprodukten zusammenarbeiten, werden die für die Zusammenarbeit benötigten Daten oder Funktionen über dafür vorgesehene Schnittstellen bereitgestellt. Stehen keine Schnittstellen zur Verfügung oder bieten diese nicht die erforderlichen Informationen, gibt es die Möglichkeit der Softwareintegration, um die Daten und Funktionen bestehender Software wiederzuverwenden.

In diesem Kapitel wird zunächst das Thema Integration von Software allgemein behandelt, bevor es ab Abschnitt 2.5 explizit um den Sonderfall der bildbasierten Integration von Software geht. In Abschnitt 2.1 erfolgt eine kurze Definition des Begriffs Softwareintegration. Im darauf folgenden Abschnitt 2.2 werden die Einsatzmöglichkeiten von Softwareintegration diskutiert. Anschließend wird in den Abschnitten 2.3 und 2.4 auf zwei wichtige Bestandteile der Softwareintegration, die Interaktion mit vorhandener Software und das Reverse Engineering eingegangen.

2.1 Definition Softwareintegration

Softwareintegration ist die Verknüpfung und Ergänzung bestehender Software zu einem neuen Softwareprodukt und lässt sich in die Schritte Interaktion und Funktion unterteilen. Die Interaktion stellt die Verbindung und die Kommunikation mit bestehender Software her. Der zweite Schritt, die Funktion, bezieht sich auf die Implementierung neuer Funktionalität.

2.2 Einsatz von Softwareintegration

Software ist in der Regel auf einen bestimmten Anwendertyp und klar definierte Aufgabenstellungen ausgelegt. Die hohen Kosten und die lange Dauer von Softwareprojekten beschränken die Rahmenbedingungen, in denen ein Softwareprodukt benutzt werden kann, auf die speziellen Anforderungen eines Benutzers oder einer Zielgruppe. Softwareintegration wird eingesetzt,

wenn Programme, unvorhergesehen, für neue Aufgaben außerhalb der ursprünglichen Spezifikationen verwendet werden sollen. Des Weiteren wird Softwareintegration für die Optimierung bewährter Arbeitsprozesse eingesetzt. Beides tritt häufig auf, wenn sich die Situation des Anwenders oder die Gruppe der Anwender verändert. In einem typischen Anwendungsfall werden mehrere Programme benötigt, um einzelne Aufgaben eines Arbeitsprozesses durchzuführen. Zur Verbesserung des Arbeitsflusses werden die vorhandenen Programme in ein neues Softwareprodukt integriert und um zusätzliche Funktionalität erweitert. Die neue Anwendung verknüpft die bisherigen Programme, indem beispielsweise die gegenseitige Steuerung und der Zugriff, auf jeweils interne Daten und Funktionen, ermöglicht werden.

2.3 Interaktion mit vorhandener Software

Es gibt unterschiedliche Wege mit vorhandener Software zu interagieren. Einige weit verbreitete Möglichkeiten werden im Folgenden vorgestellt.

2.3.1 Softwareintegration mit einem Wrapper

Mit einem Wrapper kann die vorhandene Software gestartet und wieder angehalten werden. Auf diese Weise kann ihre Funktionalität in neue Software integriert und dort erweitert werden. Ein Wrapper benutzt das zur Verfügung stehende Benutzerinterface, um mit der Originalsoftware zu kommunizieren. Möglicherweise kann durch Parameter, Tastaturkürzel, eine Kommandozeile oder Konfigurationsdateien das Verhalten der zu integrierenden Software zusätzlich gesteuert werden. Die Einsatzmöglichkeiten sind durch die Komplexität der Interaktion sowie durch das Benutzerinterface der Originalsoftware eingeschränkt. Ein wesentlicher Vorteil bei der Interaktion über einen Wrapper ist, dass die Originalsoftware nicht verändert wird. Dieser Vorteil wird relevant, wenn die Originalsoftware mit anderer Software oder sonstigen Schnittstellen verbunden ist, mit dem sie auch nach der Integration noch reibungslos funktionieren soll [13]. Beispielsweise könnte eine Verbindung mit dem Supportsystem des Herstellers, mit anderer Software oder mit einem Internetserver bestehen.

2.3.2 Integration über Hardwareschnittstellen

Weitere nichtinvasive Methoden die Funktionalität eines Programms nachträglich zu beeinflussen ergeben sich, wenn zur Interaktion nicht ausschließlich das Benutzerinterface verwendet wird. Durch das Abfangen von Datenströmen, die an bestimmte Hardwarekomponenten gerichtet sind oder über das Netzwerk geschickt werden sollen, lassen sich eventuell wichtige Informationen ermitteln. Dieselben Datenströme können auch wieder an die Software zurück geleitet werden. Häufig sind solche Datenströme allerdings

verschlüsselt oder in einem unbrauchbaren oder schwer zu interpretierendem Format. Ein Großteil der Operationen einer Software findet entweder auf dem Rechner des Anwenders oder auf einem Server statt. Nur ein Bruchteil aller Informationen wird über das Netzwerk oder zur Hardware geschickt. Ob dieser Bruchteil für eine Interaktion mit dem Programm genügt, ist situationsabhängig.

2.3.3 DLL-Injection, Binary Patching und Runtime Patching

Eine sehr mächtige und weit verbreitete Methode zum nachträglichen Manipulieren des Verhaltens von Software wird als DLL-Injection bezeichnet. Dabei wird ausgenutzt, dass viele Programme dynamische Bibliotheken zur Laufzeit laden. Solche Programme können dazu gebracht werden eine fremde DLL zu laden und auszuführen. Eine Möglichkeit eine DLL-Injection durchzuführen ist das Verwenden sogenannter Hooks. Bei Windowsanwendungen stehen dafür die Funktionen `SetWindowsHookEx` und `CreateRemoteThread` zur Verfügung. Subtilere Möglichkeiten sind das Platzieren einer eigenen DLL mit dem Namen, einer normalerweise durch das Programm aufgerufenen DLL, im Programmverzeichnis oder das Manipulieren der Windows-Registry, indem die eigene DLL als AppInit DLL angegeben wird [10]. In [4] werden weitere Techniken, wie das Binary Patching oder das Runtime Patching vorgestellt. Diese eignen sich ebenfalls dazu, Programme zu steuern ohne den Quellcode zu verändern.

Alle bisher erwähnten Methoden der Interaktion verwenden nicht den Quellcode und verursachen nur kleine Änderungen in der Originalsoftware. Besteht jedoch Zugriff auf den Quellcode der Software, kann die zur Interaktion notwendige Funktionalität einfach ergänzt oder bestehende Funktionalität extrahiert und somit eventuell auf Interaktion verzichtet werden.

2.4 Reverse Engineering

Um herauszufinden, welche Art der Interaktion sich in einem speziellen Fall am besten realisieren lässt und wie sie dann herzustellen ist, muss die zu integrierende Software untersucht werden. Das Verstehen, Lernen und Analysieren von Software wird als Reverse Engineering bezeichnet.

In dieser Arbeit ist mit Reverse Engineering, immer Software Reverse Engineering gemeint. Das ist ein Vorgang, der durchgeführt wird, um komplexe, unbekannte oder verborgene Informationen aus einem Softwaresystem zu extrahieren [1]. Beim Reverse Engineering wird die Software auseinandergenommen und komponentenweise untersucht, um ihr Design, ihre Architektur und schließlich die Funktionalität vollständig zu verstehen [15]. Diese Aussagen decken sich mit der oft zitierten Definition: „Reverse engineering is the process of analyzing a subject system to

- identify the system's components and their interrelationships and
- create representations of the system in another form or at a higher level of abstraction."[7]

Vor dem Hintergrund der Softwareintegration wird Reverse Engineering gemacht, um anschließend die verstandene Funktionalität für neue Software nutzbar zu machen. Dabei kann es auch notwendig sein, vorhandene Software so zu verändern, dass beispielsweise neue Funktionsaufrufe hinzugefügt oder alte Befehle überschrieben, beziehungsweise gelöscht werden. Beim Reverse Engineering tauchen unterschiedliche Probleme auf. Das wohl größte Problem betrifft die große Vielfalt existierender Software. Sie kann beispielsweise von verschiedenen Herstellern stammen, für bestimmte Plattformen entwickelt worden sein oder auf heterogener Softwarearchitektur aufbauen [13]. Außerdem ist Software in der Regel auf die Benutzung durch Anwender ausgelegt und verfügt über die entsprechenden Benutzeroberflächen. Diese können nicht ohne spezielle Anpassungen von einem anderen Programm interpretiert werden und liefern nur die für den Anwender relevanten Informationen. In den folgenden beiden Abschnitten wird konkreter auf die Herausforderungen der Informationsextraktion bei zugänglichem und bei nicht zugänglichem Quellcode des Originalprogramms eingegangen.

2.4.1 Interne Informationsextraktion

In seltenen Fällen besteht die Möglichkeit internen Zugriff auf die zu integrierende Software zu haben, also den originalen Quellcode benutzen zu können. Das Verhalten der Software kann dann gut nachvollzogen werden, weil die genauen Systemabläufe und eventuelle Folgen von Änderungen im Programm absehbar sind. Der Zugriff und die Wiederverwendung vorhandener Daten kann ebenfalls problemlos erfolgen, da auch die Datenstrukturen und der modulare Aufbau ersichtlich sind. Der Aufwand der Informationsextraktion richtet sich in diesem Fall lediglich nach der Qualität des vorhandenen Quellcodes, der Qualität der Dokumentation und der gewählten Technologie. Häufig führen unterschiedliche Faktoren, auch bei zugänglichem Quellcode, zu einer sehr langen Einarbeitungszeit in die alte Software. Solche Faktoren können sein, die Verwendung neuerer Technologie, zum Beispiel der Einsatz einer anderen Programmiersprache oder die Portierung auf andere Betriebssysteme, genau wie unübersichtlicher Quellcode und unvollständige Dokumentation.

2.4.2 Reverse Engineering ohne Quellcode

Anders als in [1] und [15], wo unabhängig davon, ob der Quellcode zur Softwareanalyse zur Verfügung steht oder nicht, der Begriff Reverse Engineering verwendet wird, definieren Perry und Oskov: „Reverse engineering [...] is simply the act of figuring out what software that you have no source code for

does in a particular feature or function to the degree that you can either modify this code, or reproduce it in another independent work.“ [23] Es wird also explizit gefordert, dass der Quellcode nicht verfügbar ist, was auf die allermeisten kommerziellen Produkte, aber auch auf einen Großteil der freien Software zutrifft. Die Funktionalität eines Programms zu verstehen und wiederverwendbar zu machen ist, ohne den originalen Quellcode zu kennen, eine schwere Aufgabe. Für eine umfassende Abhandlung dieser Thematik eignet sich [22] oder als Einführung [23]. An dieser Stelle sollen lediglich die Ideen zweier grundsätzlich verschiedener Ansätze, zur Herangehensweise an das Reverse Engineering ohne Quellcode, vermittelt werden.

1. Das Verhalten der vorhandenen Software bei unterschiedlichen Eingaben wird beobachtet[1]. Mögliche, für das Reverse Engineering relevante, Untersuchungsmerkmale sind zum Beispiel direkte Ausgaben über die Benutzeroberfläche, Veränderungen von Dateien auf der Festplatte, aber auch schwer zu interpretierende Merkmale, wie der verursachte Netzwerkverkehr oder der Verkehr über etwaige Hardwareschnittstellen. Das Verhalten der originalen Software wird anhand der beobachteten Merkmale simuliert und somit vorhandene Funktionalität reproduziert. Diese kann dann in neuer Software übernommen werden.
2. Durch das Disassemblieren, sowie Dekompilieren wird versucht aus einem ausführbaren Computerprogamm, zunächst Maschinencode und anschließend Quellcode, zu erzeugen. Beide Teilschritte haben große Einschränkungen was ihre Ergebnisse betrifft. Die Qualität des so erzeugten Quellcodes variiert stark. Es ist möglich, dass das Original bis auf wenige Details korrekt reproduziert wird oder auch, dass der erzeugte Quellcode völlig unbrauchbar ist. Außerdem gibt es Vorkehrungen zum Schutz vor Softwarepiraterie, die das Disassemblieren und das Dekompilieren erschweren.

2.5 Bildbasierte Integration von Software

Eine spezielle Form der Softwareintegration basiert auf einem bildbasierten Ansatz. Dabei findet die Interaktion mit vorhandener Software ausschließlich auf visueller Ebene statt. Soll eine vorhandene Software (ALT) in ein neues Softwareprodukt (NEU) integriert werden, wird ALT von NEU gestartet, während alle Informationen, die ein Benutzer über visuelle Ausgabegeräte von ALT erhalten würde, von NEU aufgenommen, verarbeitet und interpretiert werden. Auf der Grundlage dieser Informationen findet anschließend die Implementierung neuer Funktionalität statt. Das sind Tätigkeiten, wie das Treffen von Entscheidungen oder das Verfassen von Eingaben. Diese Aufgaben werden normalerweise von einem menschlichen Benutzer erledigt. Bei

[1] Dieses Vorgehen wird auch als Black-Box-Test bezeichnet.

der Integration von Software können sie an Algorithmen, andere Programme oder ebenfalls an Benutzer weitergeleitet und von ihnen erledigt werden. Benutzereingaben die ALT als Feedback erwartet, werden anschließend von NEU getätigt, indem Tastatureingaben oder Mausgesten simuliert werden. Die Interaktion bei der bildbasierten Softwareintegration erfolgt somit durch die Anwendung eines Wrappers (siehe Abs. 2.3). Für die Implementierung von Funktionalität stehen alle Möglichkeiten der Softwareentwicklung zur Verfügung.

2.5.1 Screen Scraping

Ein menschlicher Benutzer ist dazu in der Lage, die Struktur und den Inhalt einer grafischen Benutzeroberfläche (GUI), einer Webseite oder jedes anderen Bildschirminhaltes zu erfassen. Für ein Programm ist ein Bild nur eine Aneinanderreihung von Pixeln ohne jegliche Bedeutung. Beim Screen Scraping wird versucht einem Programm beizubringen, aus Bildern Informationen auszulesen und richtig zu interpretieren. Screen Scraping ist nicht neu und wird seit langem in unterschiedlichen Bereichen der Softwareentwicklung eingesetzt. Insbesondere um Altsysteme, sogenannte Legacy Systeme in neue Arbeitsumgebungen zu integrieren. Ein Legacy System ist eine Anwendung, welche sich über einen langen Zeitraum in einem Unternehmen oder einer Arbeitsumgebung etabliert hat und gegebenenfalls ständig weiterentwickelt wurde. Der Zugriff auf solche Systeme erfolgt über Benutzeroberflächen auf eigens dafür spezialisierter Hardware. Legacy Systeme sind schwer zu ersetzen, weil sie tief in den jeweiligen Arbeitsabläufen und Strukturen verankert sind. Ihre Arbeitsweise ist darüber hinaus, wegen mangelnder Dokumentation, schlecht nachzuvollziehen. Als schnelle Ad-Hoc Lösung wurde Screen Scraping eingesetzt, um trotz neuer Hard- und Softwareumgebungen nicht den Zugriff auf die wichtigen Legacysysteme zu verlieren. So wurden zunächst Kommandozeilen basierte Legacy Systeme in Systeme mit grafischen Benutzeroberflächen integriert. Dabei wird das Legacy System weiterhin eingesetzt aber von außen über eine GUI gesteuert. Obwohl die Migration von Legacysystemen bis einige Jahre nach der Jahrtausendwende ein wichtiges Forschungsgebiet war, wurde Screen Scraping, als unsaubere Zwischenlösung, von der Forschung wenig Beachtung geschenkt. Aus Mangel an Hardwareressourcen mussten an vielen Stellen Performanceoptimierungen, zu lasten der Softwaretechnik oder der Qualität der Bildverarbeitung, durchgeführt werden [9], [16]]. Später wurde Screen Scraping auch eingesetzt, um alte Benutzeroberflächen durch neue zu ersetzen oder Programme fernsteuern zu können.

2.5.2 Stand der Technik beim Screen Scraping

Ein gutes Beispiel für den heutigen Einsatz von Screen Scraping bietet die Software Sikuli Search [32]. Dabei handelt es sich um eine Suchmaschine die als Suchanfragen Bilder, anstatt Wörtern erwartet. Sikuli Search ist spezialisiert auf die Suche nach GUI Elementen. Das Programm wird eingesetzt um Benutzern das Erlernen einer neuen Software zu erleichtern, indem Screenshots von der Software gemacht werden und als Suchanfragen verwendet werden können. Die angewandten Bildverarbeitungsalgorithmen sind Template Matching und die skaleninvariante Merkmalstransformation (SIFT). Die im Rahmen dieser Arbeit entwickelte Software setzt konsequent auf Template Matching, welches in Abschnitt 5.2.1 genau erklärt wird. SIFT ist ein Algorithmus, der hauptsächlich zum Einsatz kommt, wenn der Suchraum sehr groß oder nicht klar eingeschränkt ist oder wenn die zu analysierenden Bilder groß sind. In Abschnitt 5.2 wird gezeigt, dass bei der bildbasierten Softwareintegration, wie sie hier vorgestellt wird, der Einsatz von SIFT nicht notwendig ist.

2.5.3 Vorzüge der bildbasierten Softwareintegration

Für eine bildbasierte Softwareintegration spricht das Argument, dass der damit verbundene Aufwand sehr gut einzuschätzen ist. Es ist für einen Entwickler sofort ersichtlich, welche Probleme auftreten können und ob eine bildbasierte Integration überhaupt umsetzbar ist. Anders als bei sonstigen Integrationsstrategien kann die Entwicklung beginnen, ohne fremden Programmquelltext zu lesen und ohne eine lange Einarbeitungszeit in ungewohnte Entwicklungsumgebungen. Ist kein Quelltext vorhanden und der Zugang über Netzwerk-, Hardware-, und Softwareschnittstellen erschwert oder gar unmöglich, so ergibt sich ein weiterer Vorteil der bildbasierten Softwareintegration. Nämlich, dass gar keine andere Möglichkeit besteht die Software zu integrieren. Ein weiteres Argument ist, dass die vorhandene Software nicht verändert werden muss. Sie kann weiterhin in ihrer normalen Umgebung laufen und muss im Idealfall nicht einmal neu gestartet werden. Auch die Fernsteuerung der alten Software ist bei einer bildbasierten Softwareintegration unproblematisch, da nur Bildschirminformationen und Maus- bzw. Tastatureingaben übertragen werden müssen.

2.5.4 Nachteile der bildbasierten Softwareintegration

Die Liste der Nachteile wurde früher durch das Argument „schlechte Performance“ angeführt. In Zeiten nahezu unbegrenzter Hardwareressourcen spielt die Performance keine so zentrale Rolle mehr. Es lässt sich allerdings nicht verleugnen, dass in performance-kritischen Anwendungen bildbasierte Algorithmen, sowie die Steuerung über simulierte Maus- und Tastatureingaben schnell zum Flaschenhals werden können und daher eher ungeeignet sind.

Auch der Entwicklungsaufwand kann enorm sein, wenn die Software, die zu integrieren ist eine sehr komplexe, dynamische oder häufig wechselnde Benutzeroberfläche aufweist. In vielen Bereichen kann es sein, dass die bildbasierte Softwareintegration nicht möglich oder nicht sinnvoll ist.

Kapitel 3

Einführung in die Domäne Poker

In diesem Kapitel werden wichtige Grundlagen des Pokerspiels vermittelt, bevor es ab Kapitel 4 um die Implementierung eines Expertensystems für Online-Poker-Plattformen geht. Dazu wird in Abs. 3.1 umfangreich auf die Terminologie und das Regelwerk des Spiels eingegangen, bevor ausgewählte strategische Konzepte eingeführt werden. Im darauf folgenden Abschnitt 3.3 wird das Thema Online-Poker behandelt, indem die Gegebenheiten der Pokerplattformen, aus der Perspektive der Softwareintegration, analysiert werden. Im letzten Abschnitt dieses Kapitels (Abs. 3.4) werden vorhandene Expertensysteme und Pokersoftware vorgestellt und verglichen.

3.1 Das Pokerspiel

Poker ist ein weit verbreitetes Kartenspiel mit vielen Varianten. Bei einem Pokerspiel setzen die Spieler, in mehreren Wettrunden, Einsätze auf die Gewinnchancen ihrer Karten. Die Spieler entscheiden, ob sie die Einsätze der anderen Spieler mitgehen oder aus dem Spiel aussteigen. Die gesamten Einsätze gewinnt, wer als Letzter im Spiel geblieben ist oder wer am Ende die besten Karten hat. Da die Karten der anderen Spieler unbekannt sind, zählt Poker zu den Spielen mit unvollständiger Information. Das Spiel wird in staatlichen Spielbanken, auf privaten Turnieren und in großem Umfang auch online gespielt. Obwohl Poker rechtlich als Glücksspiel eingestuft wird, kann der Spielverlauf durch strategische, mathematische und psychologische Maßnahmen beeinflusst werden. Die einheitliche Meinung von Experten besagt, dass Poker eine Mischform aus Glücksspiel und Geschicklichkeitsspiel ist, wobei die Glückskomponente mit der Anzahl der gespielten Spiele sinkt, während die Geschicklichkeitskomponente ansteigt [12].

Beim Poker werden sehr viele Fachbegriffe benutzt, diese werden in den kommenden Abschnitten kontinuierlich eingeführt. Aus dem englischen ent-

liehene Pokerbegriffe werden nur benutzt, wenn kein deutsches Äquivalent geläufig ist.

3.1.1 Spielvarianten

Es gibt sehr viele Pokervarianten. Die mit Abstand am weitesten verbreitete Variante nennt sich *Texas Hold'em* und kann in den Modi *No Limit*(NL), *Fixed Limit*(FL) oder *Pot Limit*(PL) gespielt werden. Die drei Modi unterscheiden sich nur darin, dass die erlaubte Anzahl an Chips[1] die pro Spielzug gesetzt werden dürfen nicht beschränkt NL, genau festgelegt FL oder durch eine variable Obergrenze limitiert PL ist. In dieser Arbeit werden ausschließlich die Texas Hold'em FL Pokerregeln verwendet. Aus Gründen der Lesbarkeit wird im Folgenden nur noch der Oberbegriff Poker benutzt.

3.1.2 Regeln

In diesem Abschnitt geht es erst einmal darum, das Spiel Poker in seinen Grundzügen zu vermitteln. Dabei werden bewusst die Regeln vereinfacht und viele Details vorenthalten. An geeigneter Stelle werden im weiteren Verlauf dieser Arbeit Ergänzungen, zu den hier eingeführten Regeln, gemacht. Gespielt wird mit einem regulären Kartenspiel aus 52 Karten. Jede Karte hat einen *Wert* [2, 3, 4,..., 10(T), Bube(J), Dame(Q), König(K), Ass(A)] und eine *Farbe* [Herz(♡), Pik(♠), Caro(♢), Kreuz(♣)]. Es spielen zwei bis zehn Spieler an einem Tisch. Ziel des *Spiels* ist es, seinen Einsatz zu vervielfältigen, indem *Hände* gewonnen werden. Eine Hand beginnt, nachdem die Grundeinsätze[2] gesetzt wurden und bezeichnet ein vom vorherigen Spielverlauf unabhängiges Teilspiel. Es können vor jeder Hand neue Spieler beitreten und bereits teilnehmende Spieler aussetzen. Eine Hand besteht aus maximal vier *Phasen* [*Preflop*, *Flop*, *Turn*, *River*] und diese wiederum aus mehreren *Wettrunden*. In jeder Wettrunde haben alle Spieler die Möglichkeit Einsätze zu machen. Getätigte Einsätze befinden sich im *Pot*. Der Gewinner einer Hand bekommt den Pot.

Mögliche Aktionen

Beim Poker geht es darum, Wetten auf die eigenen Gewinnchancen zu machen. Die Aktionen die dafür zur Verfügung stehen sind:

- *Aussteigen* - Entscheidet sich ein Spieler zum Aussteigen, kann er während der aktuellen Hand nicht wieder in das Spiel eintreten. Er gibt seine Karten ab und hat keine Möglichkeit mehr den Pot zu gewinnen. Seine bisherigen Einsätze verbleiben im Pot.

[1] Ein Chip ist eine wertneutrale Einheit für Einsätze.

[2] Die Grundeinsätze werden Blinds genannt. Es gibt einen Big Blind und einen Small Blind.

- *Schieben, Mitgehen* - Damit ein Spieler im Spiel bleiben kann, muss er den Einsatz der anderen Spieler mitgehen. Sofern noch keine Einsätze gemacht wurden, kann geschoben werden.
- *Erhöhen, Setzen* - Gegnerische Einsätze können erhöht werden, oder es kann -sofern noch keine Einsätze gemacht wurden- ein neuer Einsatz gemacht werden. Bei letzterem wird der Begriff *Setzen* verwendet. Bei Limit Poker ist der Betrag einer Erhöhung festgelegt und abhängig von der aktuellen Phase. Eine Erhöhung im Preflop[3] oder Flop entspricht dabei dem Betrag eines *Small Blind*, in den zwei folgenden Phasen, dem eines *Big Blind*.

Ablauf einer Hand

Zu Beginn einer Hand setzen die ersten zwei Spieler links vom Kartengeber festgelegte Grundeinsätze. Der erste Spieler setzt den Small Blind, der zweite den etwa doppelt so großen Big Blind. Sobald die *Blinds* gesetzt wurden, verteilt der Kartengeber je zwei Karten an jeden Spieler. Jede Wettrunde startet bei dem ersten Spieler links vom Kartengeber. Im Uhrzeigersinn hat jeder Spieler die Möglichkeit eine der vorgestellten Aktionen zu machen. Eine Ausnahme ist dabei die erste Runde einer Hand, bei welcher die ersten zwei Spieler verpflichtet sind ihre Einsätze blind zu setzten. Eine neue Wettrunde beginnt immer, wenn eine Erhöhung stattfindet. Je Phase sind maximal vier Wettrunden möglich. Eine neue Phase beginnt, wenn alle im Spiel verbliebenen Spieler in der aktuellen Phase mindestens einmal an der Reihe waren und den selben Einsatz im Pot haben. Eine neue Phase ist automatisch eine neue Wettrunde. Zusätzlich zu den zwei *Handkarten*, werden im Verlauf einer Hand insgesamt fünf weitere Karten, die sogenannten *Gemeinschaftkarten*, für alle sichtbar in die Mitte des Tisches gelegt. Es werden drei Karten im Flop und jeweils eine Karte im Turn und River aufgedeckt. Am Ende der letzten Phase kommt es zum *Showdown*, wenn mindestens zwei Spieler nach der letzten Wettrunde noch nicht ausgestiegen sind. Beim Showdown werden die Karten, der noch im Spiel verbliebenen Spieler, nacheinander aufgedeckt.

Gewinnen einer Hand

Ein Spieler gewinnt eine Hand, wenn entweder alle Mitspieler ausgestiegen sind oder er beim Showdown das stärkste *Blatt* hält. Ein Blatt besteht immer aus genau fünf Karten und kann beliebig aus den Gemeinschaftskarten und den Handkarten kombiniert werden. Tabelle 3.1 zeigt alle möglichen Blätter nach ihrer Wertigkeit sortiert. Haben mehrere Spieler das selbe Blatt, so entscheiden weitere Kriterien darüber, welches Blatt stärker ist. Falls beim Showdown mehrere Spieler ein Paar haben, so gewinnt der derjenige, der das Paar hat, dessen Kartenwert am höchsten ist. Haben mehrere Spieler

[3] Es ist geläufig zu sagen: „im flop“, statt: „während der Flopphase“.

das selbe Paar, so entscheidet ein so genannter *Kicker*. Der Kicker ist die höchste Karte, die ein Spieler neben dem eigentlichen Blatt hat. Ist auch der Kicker bei mehreren Spielern gleich, so entscheidet die zweithöchste Karte und so weiter. Genauso verhält es sich bei Drillingen und Vierlingen. Bei mehreren Spielern mit einem Doppelpaar gewinnt das Doppelpaar, welches das höchste Einzelpaar hat. Ist dieses gleich, so entscheidet das zweite Paar und anschließend wieder der Kicker. Bei einem Full House verhält es sich ähnlich. Zunächst werden die Drilling verglichen, dann das Paar. Bei den übrigen Blättern Straße, Flush und Straight Flush gewinnt der Spieler, der die höchste Einzelkarte hält. Alle vier Farben sind gleichwertig. Haben weiterhin mehrere Spieler gleichstarke Blätter, so kommt es zu einem Split Pot[4].

Tabelle 3.1: Poker Blätter

Blatt	**Beschreibung**
Höchste Karte	Karte mit der höchsten Wertigkeit.
Paar	Zwei Karten derselben Wertigkeit.
Doppelpaar	Zwei Paare.
Drilling	Drei Karten derselben Wertigkeit.
Straße	Fünf aufeinander folgende Karten.
Flush	Fünf Karten derselben Farbe.
Full House	Drilling und Pärchen.
Vierling	Vier Karten derselben Wertigkeit.
Straight Flush	Straße in gleicher Farbe.
Royal Flush	Höchst mögliche Straße in gleicher Farbe.

3.1.3 Pokerstrategie

In der deutschen Rechtsprechung heißt es „Poker ist überwiegend zufallsbezogen und somit ein Glücksspiel“ [3]. Im Widerspruch dazu steht, dass es viele Spieler gibt, welche langfristig und konstant, durch Poker spielen sehr viel Geld verdienen [19]. Der Faktor Glück ist besonders relevant, wenn nur ein kleines Zeitfenster betrachtet wird. Langfristig zahlt sich eine auf mathematischen und psychologischen Grundlagen basierende Spielstrategie aus. Wie in allen Spielen, welche einen Glücksfaktor beinhalten, ist ein positiver Erwartungswert erstrebenswert. In Glücksspielen wie *Roulette* oder *Black-Jack* ist der Erwartungswert konstant und für den Spieler negativ. Im Poker hingegen kann dieser durch eine gute Strategie positiv beeinflusst werden, wodurch ein langfristiger Gewinn möglich wird.

In diesem Abschnitt werden die Grundlagen der Pokerstrategie behandelt und dabei wichtige Begriffe eingeführt.

[4] Der Pot wird unter den Spielern, die das höchste Blatt haben, zu gleichen Teilen aufgeteilt.

Preflop Verhalten

Das Preflop Verhalten stellt das Fundament für eine erfolgreiche Strategie. In jeder Literatur zu Pokerstrategien lassen sich Anweisungen für eine empfehlenswerte Spielweise im Preflop finden. Dabei werden nur überdurchschnittlich starke Handkarten gespielt. Ein typischer Anfängerfehler ist es, zu viele Handkarten zu spielen. Dadurch setzt der Spieler langfristig zu viel Geld auf Handkarten, bei denen die Wahrscheinlichkeit im Showdown das beste Blatt zu halten, zu gering ist. Starke Handkarten zeichnen sich dadurch aus, dass sie entweder schon ein gutes Blatt, in Form von einem Paar, sind oder die Wahrscheinlichkeit hoch ist, dass ein gutes Blatt mit den Gemeinschaftskarten gebildet werden kann. Dabei ist es zum Vorteil, wenn die Handkarten mindestens eine der Eigenschaften aus Tabelle 3.2 haben.

Tabelle 3.2: Eigenschaften von Preflopblättern

Bezeichnung	**Eigenschaft**	**Beispiel**
Paar	dieselbe Wertigkeit	A ♡, A ♢
Suited	dieselbe Farbe	J ♠, 9 ♠
Connected	die Wertigkeiten folgen aufeinander	10 ♢, J ♣
Broadway	die Wertigkeiten sind mindestens 10	A ♣, Q ♡

Spielweise

Im Poker wird zwischen lockeren und strengen Spielern unterschieden. Lockere Spieler gehen viele Starthände mit und hoffen auf vorteilhafte Gemeinschaftskarten. Diese Spielweise hat einen negativen Erwartungswert und kann auf Dauer nicht gewinnbringend sein. Strenge Spieler hingegen spielen ausschließlich Hände mit positivem Erwartungswert. Dabei zahlt sich die längere Wartezeit auf vorteilhafte Starthände und erfolgsversprechende Gemeinschaftskarten langfristig aus. Weitere Unterschiede lassen sich im Setzverhalten finden. Aggressive Spieler setzen Beträge, sobald ein gutes Blatt getroffen wurde. Passive Spieler hingegen gehen hauptsächlich bei bereits gesetzten Einsätzen mit und setzen selten einen Betrag. Die aggresive Spielweise ist hierbei empfehlenswerter, da Spieler mit einem unvollständigen Blatt aus der Hand aussteigen, bevor diese vervollständigt werden kann.

Position

Die Position des Spielers wirkt sich auf die Entscheidungsfindung aus. Die Auswirkung besteht darin, dass eine Entscheidung in Abhängigkeit von den Aktionen der Mitspieler getroffen wird. Die Spieler, welche zuerst an der Reihe sind, verfügen über weniger Informationen als die Mitspieler, die erst später in einer Wettrunde eine Aktion tätigen müssen. An einem üblichen

Abbildung 3.1: Positionen - Blau sind die beiden Blinds, rot die frühen, Gelb die mittleren und grün die späten Positionen.

Pokertisch für zehn Spieler werden die Sitzpositionen in vier verschiedene Gruppen unterteilt. Es wird zwischen *Blind*, *früher*, *mittlerer* und *später Position* unterschieden. Dabei ist die Position des Spielers von der Entfernung zum Kartengeber abhängig.

Die Position spielt auch für das Preflop Verhalten eine wichtige Rolle. Beim Spielen in einer frühen Position besteht das Risiko, dass ein gesetzter Betrag von einem späteren Mitspieler erhöht wird und so ein hoher Betrag für das Erreichen der nächsten Phase gesetzt werden muss. Aus diesem Grund werden in früheren Positionen weniger Karten gespielt als in späteren. In späteren Spielphasen bleibt die Position bestehen. Es muss wieder eine Entscheidung ohne die Informationen über die Mitspieler getroffen werden. Ergibt sich für den Spieler kein gutes Blatt durch die Gemeinschaftskarten, setzt er keinen Betrag und schiebt. Das Schieben signalisiert den Mitspielern, dass der Spieler kein gutes Blatt hat. Dieser Informationsvorteil kann von den Mitspielern ausgenutzt werden, indem sie einen Betrag setzten.

Outs

Häufig kommt ein Spieler in die Situation, dass ihm nur eine Karte fehlt, um ein Blatt zu vervollständigen. Dieser Fall wird *draw* genannt. Sollten noch nicht alle Gemeinschaftskarten gelegt worden sein, bleibt eine Restchance, dass diese Karte noch gelegt wird. Meistens kann das Blatt durch mehrere unterschiedliche Karten vervollständigt werden. Diese Karten werden *Outs* genannt. Fehlt beispielsweise nur noch eine Karte um einen Flush zu bilden, gibt es neun Outs, da es insgesamt 13 verschiedene Karten einer Farbe im Kartendeck sind und vier davon bereits verwendet werden.

Odds

Die Wahrscheinlichkeit das Blatt durch die fehlenden Outs zu vervollständigen, wird, nach dem englischen Wort für Chance, *odd* genannt. Es ist zu beachten, dass das Vervollständigen einer Hand nicht unbedingt zu einem Sieg führt, da sich oftmals höhere Blätter mit den Gemeinschaftskarten bilden lassen. Die Wahrscheinlichkeit, dass ein Out in der kommenden Phase aufgedeckt wird, lässt sich leicht berechnen. Dabei wird die Anzahl der Outs durch die Anzahl der unbekannten Karten geteilt.

$$\frac{Outs}{unbekannteKarten} = Odds$$

Anhand des Beispiels aus dem vorherigem Abschnitt wird angenommen, dass dem Spieler neun Outs zum Vervollständigen eines Blattes fehlen. Die aktuelle Phase sei Flop, sodass bereits drei Gemeinschaftskarten gelegt worden sind. So befinden sich weitere 47 unbekannte Karten im Spiel. Die Wahrscheinlichkeit, dass eine der neun Karten aufgedeckt wird errechnet sich durch $\frac{9}{47} \approx 0,19$. Um die Rechnung zu vereinfachen, bedienen Spieler sich häufig der groben Faustformel $outs * 2 + 2$. Die Formel liefert ein ungenaues Ergebnis, ist für die meisten Fälle dennoch ausreichend.

Pot Odds

Die Pot Odds beschreiben das Verhältnis von einem möglichen Gewinn, zu einem zu setzenden Betrag. Es gilt, dass der zu setzende Betrag durch den möglichen Gesamtgewinn geteilt wird. Der Gesamtgewinn ergibt sich aus $Pot + Einsatz$.

$$\frac{zusetzenderBetrag}{Gesamtgewinn} = PotOdds$$

Durch das errechnen der Pot Odds und Odds kann ein Spieler feststellen, ob es sich lohnt eine unvollständige Hand bei einer Erhöhung mitzugehen. Dabei muss der Wert der Odds größer sein als der Wert der Pot Odds. Wenn die Wahrscheinlichkeit auf ein vollständiges Blatt größer ist, als das Verhältnis des Einsatzes zum Gewinn, so ist auch der Erwartungswert dieser Entscheidung positiv.

Slow Play

Slow Play ist eine Taktik, die eingesetzt werden kann, wenn bereits in einer frühen Phase ein sehr hohes Blatt getroffen wurde. Dabei wird geschoben, um ein schlechtes Blatt vorzutäuschen. Sobald ein Mitspieler einen Einsatz gesetzt hat, wird dieser wiederrum erhöht. Durch diese Taktik wird die eigentliche Spielweise verschleiert und ist schwieriger zu durchschauen. Außerdem ist es möglich dadurch Gegner zu provozieren mehr Geld in den Pot zu legen, wenn das eigene Blatt nicht mehr zu übertreffen ist.

Hand schützen

Diese Technik beruht darauf, auf ein bereits vollständiges, wenn auch nicht hochwertiges, Blatt zu setzen, so lange noch nicht alle Gemeinschaftskarten aufgedeckt wurden. Dabei wird versucht, die Mitspieler zum Aussteigen zu zwingen. Halten die anderen Mitspieler einen Draw mit wenig Outs, so würden sie beim Mitgehen einen negativen Erwartungswert haben.

Gewinnwahrscheinlichkeit

Diese Berechnung kann ein Spieler in dieser Form nicht anwenden oder gebrauchen. Bei Fernsehübertragungen von großen Poker-Events wird die Wahrscheinlichkeit der einzelnen Spieler für einen Gewinn im Showdown angezeigt. Dabei müssen die Handkarten der Mitspieler bekannt sein, was für Spieler am Tisch unmöglich ist. Der hier benutzte Algorithmus testet die bekannten Blätter auf die noch möglichen Kombinationen der unaufgedeckten Karten. Je öfter ein Blatt bei den möglichen Kombinationen gegen die restlichen Gewinnt, desto höher wird natürlich die Wahrscheinlichkeit auf einen Gewinn im Showdown. Für einen Spieler ist der Ansatz von diesem Algorithmus dennoch interessant, da oft eingeschätzt werden muss, wie viele Starthände, bei gegebenen Gemeinschaftskarten, stärker als das eigene Blatt sind und ob sich die Situation des Gegners im Laufe der Hand durch bestimmte Karten verbessern kann.

Psychologische Komponente

Weit verbreitete Klischees zeigen Poker als ein Spiel bei dem sehr viel mentale Stärke notwendig ist. Diese Klischees sind ein wenig überzogen, dennoch bieten psychologische Tricks dem Spieler, der sie beherrscht, einige Vorteile. Insbesondere ist das genaue Beobachten der Mitspieler von großer Bedeutung. Oft lassen Mimik oder Gestik Schlüsse auf die Qualität der gegnerischen Karten zu. Der Umgang mit so gewonnenen Informationen sollte allerdings sehr vorsichtig sein, da auch falsche Signale versendet werden können, um den Gegner zu verwirren.

Bluffen

Eine aggressive Spielweise erfordert es manchmal Chips zu setzen, obwohl das eigene Blatt nicht sehr stark ist. Es kommt auch vor, dass einem gegnerischem Spieler keine guten Karten zutraut werden und erhöht wird, in der Annahme, dass der Gegner nicht mitgeht. Beide Fälle können als Bluff bezeichnet werden.

3.2 Anleitung nach Pokerstrategy.com

Auf der Webseite `www.Pokerstrategy.com` finden sich Anleitungen, nach denen es möglich sein soll auf lange Sicht konstant Gewinne zu erzielen. Für die Variante *Fixed Limit Holdem* heisst es auf der Webseite, *dass man für diese Spielart schnell eine funktionierende und profitable Basisstrategie erlernen kann*[24].

Der Betreiber der Webseite überweist jedem Spieler, der sich bereit erklärt, nach den Anleitungen zu spielen ein Startgeld von 50$, um anschließend an den Gewinnen mitzuverdienen.

3.2.1 Das Spiel vor dem Flop

Für das Spiel vor dem Flop werden die Handkarten in folgende Kategorien eingeteilt:

- *Sehr starke Hände:* AA, KK, QQ / AKs, AKo [5]
- *Starke Hände:* JJ, TT, 99 / AQs, AQo, AJs
- *Mittelstarke Hände:* AJo, ATs, ATo, KQs, KQo
- *Starke spekulative Hände:* 88 - 22 / KJs, KTs, QJs, QTs, JTs, T9s
- *Gemischte Hände:* KJo, KTo, QJo, QTo, JTo / A9s -A2s, K9s, 87s, 98s

Für jede Kategorie gibt es eine eigene Tabelle mit den Aktionen der Gegner auf der einen Achse und der eigenen Position auf der Anderen. Diesen Tabellen ist zu entnehmen, welche Aktion in der gegebenen Situation durchzuführen ist. Tabelle 3.3 dient als Beispiel für solch eine Tabelle[6]. Durch die Tabellen nicht berücksichtigt wird der Fall, dass mehr als ein Mitspieler vorher erhöht hat. Für diesen Fall gibt es eine Sonderregelung: *Du legst alle Karten weg bis auf AA, KK, QQ, AKs und AKo. Mit diesen Händen erhöhst du weiter* [25]. Ähnlich formulierte Regeln gibt es für die Fälle, in denen die Prefloppphase mehrere Wettrunden hat.

Tabelle 3.3: Tabelle, nach der vor dem Flop gespielt werden soll, wenn man eine Mittelstarke Hand hat.

Aktionen der Gegner	**Frühe Pos.**	**Mittlere**	**Späte**	**SB**	**BB**
Alle steigen aus.	Aussteigen		Erhöhen		
Einer geht mit.	Aussteigen		Erhöhen		
Zwei oder mehr mit.	Aussteigen		Erhöhen		
Einer erhöht, Keiner mit.	Aussteigen				Mitgehen
Einer erhöht, Einer mit.	Aussteigen (KQs mitgehen)				Mitgehen

[5] Spielt die Farbe der Karte keine Rolle wird sie bei der Notation häufig weg gelassen. Um zu verdeutlichen, dass zwei Karten die selbe Farbe haben wird ein *o* angehängt, ansonsten ein *s*.

[6] Weitere Tabellen und eine Zusammenfassung der Strategie finden sich in [25].

3.2.2 Das Spiel nach dem Flop

Nachdem Gemeinschaftskarten in der Mitte liegen, wird das Spiel komplexer. Zum größten Teil liegt das daran, dass es wesentlich mehr verschiedene Blätter gibt, die gebildet werden können. Pokerstrategy unterscheidet zusätzlich zu den in Tabelle 3.1 aufgeführten Blättern folgende Kategorien:

- *Top-Paar* - Ein Paar aus einer Handkarte und der höhsten Gemeinschaftskarte.
- *Überpaar* - Ein Paar aus Handkarten, welches höher ist alle Gemeinschaftskarten.
- *OESD*[7] - Nach unten und oben offene Straße aus vier Karten.
- *Flushdraw* - Vier Karten einer Farbe.
- *Monsterdraw* - OESD + Flushdraw.
- *Gutshot* - Unvollendete Straße, bei der eine Karte aus der Mitte zum Vervollständigen fehlt.
- *Doppelter Gutshot* - Unvollendete Straße, bei der zwei Karten zum Vervollständigen helfen.
- *Überkarten* - Beide Handkarten sind größer als alle Gemeinschaftskarten.
- *Gutshot + Überkarten*

Für jede Phase ab dem Flop gibt es ausformulierte Regeln, die je nach Situation bestimmte Aktionen empfehlen. Die Regeln für die Flopphase sind nach Blättern sortiert und sehen folgendermaßen aus:

Top-Paar oder Überpaar

- Du setzt oder erhöhst einmal.
- Wird nach dir erhöht, gehst du nur noch mit

[7] Open-End-Straight-Draw

Auch kompliziertere Regeln treten auf wie beispielsweise:

Gutshot oder Überkarten

- Du setzt nur, wenn du vor dem Flop erhöht hast und maximal zwei Gegner hast.
- Du gehst einen gegnerischen Einsatz oder eine Erhöhung nur mit, wenn der Pot mindestens das Zehnfache des zu bringenden Einsatzes enthält

Ab dem Turn werden die Regeln nach den Aktionen der Mitspieler gruppiert und anders Formuliert:

Es hat vor dir schon jemand gesetzt:

- du erhöhst mit ...
  - Zwei Paaren oder besser.
- du gehst mit mit...
  - Top-Paaren oder Überpaaren.
  - Monsterdraws, Flushdraws, OESDs oder doppelten Gutshots.
  - Gutshots, wenn der Pot mindestens das zehnfache des von dir zu zahlenden Einsatzes enthält.
- du steigst aus mit ...
  - Jeder anderen Hand

3.2.3 Diskussion der Anleitung

Generell ist zu sagen, dass nach der Anleitung von www.Pokerstrategy.com nur sehr wenige Starthände gespielt werden und die Position des Spielers einen großen Einfluss auf die Spielstrategie hat. Diese Vorgehensweise deckt sich mit den Erkenntnissen, die eine bekannte Pokerplattform gewonnen hat. Es wurden 122 Millionen Hände aufgezeichnet, um den Erwartungswert verschiedener Starthände zu ermitteln. Die Aufzeichnung erfolgte auf Echtgeldtischen. Durch die Gewinne und Verluste, die bei den jeweiligen Starthänden erzielt werden, kann der Erwartungswert empirisch angenähert werden. Der Erwartungswert für eine bestimmte Starthand kann nicht rein rechnerisch bestimmt werden, weil der Einfluss der zahlreichen Komponenten nicht eindeutig ist. Es wurde ermittelt, dass nur die besten 40 Starthände einen positiven Erwartungswert haben. Dabei haben die besten drei Starthände einen Erwartungswert, welcher über dem Einsatz des Big Blinds ist. Weiterhin fällt auf, dass der Erwartungswert höher wird, je später die Position eines Spielers ist. Von daher werden in späteren Positionen mehr Handkarten gespielt, als in den Frühen [27]. Die Rolle der Position wurde bereits in Abschnitt 3.1.3 näher erläutert.

3.3 Online-Poker

Online-Poker unterscheidet sich in einigen Punkten von traditionellem Poker, bei dem alle Spieler gemeinsam an einem Tisch sitzen. Es ist beispielsweise nicht möglich, an der Gestik eines Mitspielers zu erkennen, ob er gute Karten hat. Es ist außerdem möglich an vielen Spielen gleichzeitig teilzunehmen, was dazu führt, dass Spieler einerseits weniger konzentriert, andererseits aber viel geduldiger handeln.

Es gibt eine Vielzahl unterschiedlicher Pokerplattformen im Internet. Die Unterschiede erstrecken sich auf der Darstellungsebene von den Bedienelementen, über die Avatare der Spieler, bis hin zu wesentlichen Spielelementen wie den Karten, den Jetons oder dem gesamten Pokertisch. Auch spielerisch gibt es Abweichungen zwischen den verschiedenen Plattformen. So bieten manche Anbieter zum Beispiel ausgefallene Pokervarianten oder Tische mit einer erhöhten oder verringerten Anzahl an Spielern pro Tisch. Für eine stabile, bildbasierte Informationsgewinnung, als Basis der bildbasierten Softwareintegration müssen alle Facetten des jeweiligen Pokerraums bekannt sein. Solch eine Analyse kann derzeit nicht automatisch, sondern muss manuell durchgeführt werden und ist aufwändig. Um die Integration des Expertensystems dennoch nicht auf nur eine Pokerplattform beschränken zu müssen, bietet es sich an, die Ähnlichkeiten unterschiedlicher Anbieter herauszuarbeiten und die Analyse somit systematisch und effizient durchzuführen. In den folgenden Abschnitten wird auf den allgemeinen Ablauf (Abs.3.3.1) und die Darstellung (Abs. 3.3.2) von Online-Poker eingegangen.

3.3.1 Ablauf eines Onlinespiels

Ohne Berücksichtigung besonderer und für diese Arbeit nicht relevanter, Ausnahmen, lassen sich folgende Abläufe auf allen gängigen online Pokerplattformen beobachten. Nach dem Betreten eines Spielraums wird ein freier Platz durch einen Klick auf denselben eingenommen. Sobald eine neue Runde beginnt, hat der beigetretene Spieler sofort die Möglichkeit mitzuspielen. Die Interaktion mit dem Spiel wird durch Klicken auf einfache Bedienelemente durchgeführt. Je nach gerade möglichen Aktionen stehen dem Benutzer bis zu fünf Schaltflächen zur Auswahl. Ist der Spieler nicht am Zug werden demnach auch keine Schaltflächen angezeigt. Der Spieler trifft seine Auswahl anhand aller ihm zugänglichen Informationen und klickt auf die entsprechende Schaltfläche. Dafür hat der Spieler, abhängig von dem Spielmodus, zwischen zehn und 60 Sekunden Zeit. Ist eine Runde beendet wird der Pot dem Kontostand des Gewinners gutgeschrieben.

Abbildung 3.2: Screenshot eines Pokertisches.

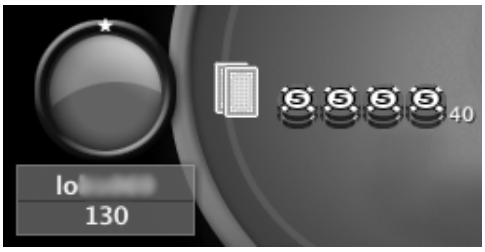
3.3.2 Grafische Darstellung von Pokerplattformen

Anhand der Abbildung 3.2 werden verschiedene Darstellungen von wichtigen Spielinformationen durch gekennzeichnete Bereiche hervorgehoben. Während einer Spielrunde werden die Handkarten des Spielers (A), sein Kontostand (B), der Pot (C), die Gemeinschaftskarten (D), die Position des Kartengebers (E) sowie alle Kontostände der anderen Mitspieler (F) angezeigt. Die Handkarten sowie der eigene Kontostand sind meistens unter dem eignen Avatar zu sehen 3.3. Der Pot und die Gemeinschaftskarten sind in der Tischmitte untergebracht 3.4. Die Aktionen der Mitspieler werden oft durch ein relativ kurzes Aufblinken der getätigten Aktion unterhalb des Avatares angezeigt. Es ist aber auch permanent zu sehen, welche Spieler noch im Spiel sind. Bei Spielern, die aus dem Spiel ausgestiegen sind, werden keine verdeckten Handkarten mehr eingeblendet. In Abbildung 3.2 befindet sich eine beispielhafte Darstellung von verdeckten Handkarten im Bereich (G). Haben Spieler in der aktuellen Bietrunde einen Einsatz getätigt oder sind einen Einsatz mitgegangen, so wird das durch Jetons auf dem Tisch unmittelbar vor ihrem Avatar kenntlich gemacht (3.5). Auf dieselbe Weise wird auch der Pot in der Tischmitte dargestellt. Es ist möglich aus der Anzahl und der Farbe der Jetons den Wert eines Jetonhaufens zu bestimmen. Dieser Wert steht jedoch auch immer in kleiner Schrift darunter. Die Position des Kartengebers und somit auch der Blinds, wird durch einen Dealerbutton auf dem Tisch vor einem der Spieler angezeigt (3.2 (E)).

Abbildung 3.3: Eigene Karten und Kontostand.

Abbildung 3.4: Gemeinschaftskarten und Pot.

Abbildung 3.5: Einsatz von Jetons.



3.4 Pokersoftware

Die Popularität von Poker und die Tatsache, dass die Komponenten Glück, Strategie und Psychologie wesentliche Bestandteile des Spiels sind, begründen, dass Poker in einer Vielzahl von Forschungsfeldern auf unterschiedliche Weise untersucht wird, wodurch eine große Anzahl unterschiedlicher Software zu diesem Thema existiert. Es gibt mathematische Programme zur Berechnung von Gewinnchancen, Software zur statistischen Analyse der Mitspieler, Expertensysteme, die mittels künstlicher Intelligenz (KI) selbstständig Poker spielen, Pokerplattformen, auf denen Spieler aus der ganzen Welt gegen-

einander spielen können und eine ganze Reihe anderer Anwendungen. Aufgrund dieser Vielfalt von Software, die eine Thematik aus unterschiedlichen Perspektiven behandelt und somit voneinander profitieren könnte, besteht der Wunsch nach Softwareintegration und bietet daher die Möglichkeit, den Ansatz der bildbasierten Integration von Software an einem realistischen, forschungsnahem und aktuellem Beispiel zu untersuchen.

3.4.1 Wissenschaftliche Pokerbots

Pokerbots sind Programme, die selbstständig Poker spielen. Die Funktionen und die Einsatzgebiete verschiedener Pokerbots sind sehr unterschiedlich. Im Wesentlichen sind zwei Grundformen zu unterscheiden: (1) Wissenschaftliche Programme mit Schwerpunkt auf der Künstlichen Intelligenz und (2) Programme die auf Pokerplattformen um echtes Geld spielen.

Wissenschaftliche Pokerbots haben im Gegensatz zu den Programmen, die auf echten Pokerplattformen spielen eine große Hürde weniger zu überwinden. Sie spielen in einem eingeschränkten, speziell auf Computerspieler ausgelegten Umfeld. Die Pokerplattform stellt dem Pokerbot eine Schnittstelle zur Verfügung über die alle relevanten Informationen über das aktuelle Spiel ausgelesen werden können. Dazu zählen beispielsweise die Aktionen der Gegner, die Karten und die Beträge, die gesetzt wurden. Mithilfe dieser Informationen entscheidet die Künstliche Intelligenz des Pokerbots, welche Aktion als nächstes gemacht wird und kann diese Entscheidung an die Plattform weiterleiten.

- Die Programme Loki [21] und sein Nachfolger Poki [8] sind zwei der ersten Pokerbots, die in einem wissenschaftlichen Umfeld entstanden sind. Poki verwendet während der Prefloppphase eine Starthandtabelle zur Aktionsbestimmung. In den späteren Phasen wird mit Hilfe von Metriken die Stärke und das Potential der eigenen Hand berechnet und mit allen anderen möglichen Händen ins Verhältnis gesetzt. Zum Reduzieren des Suchraums wird eine Form des Monte Carlo Algorithmus (vergl. Abs. 5.3.3) verwendet. Außerdem versucht Poki seine Gegner in Kategorien einzuteilen, um so Zusatzinformationen zu erhalten.
- Der Pokerbot BRPlayer[29] verfolgt eine ganz andere Strategie. Er zählt welcher Spieler, wie oft, welche Aktion gemacht hat und versucht so, die nächsten Züge vorherzusagen. BRPlayer entscheidet abhängig von diesen Vorhersagen, welche Aktion er macht. Bemerkenswert ist, dass beim Zählen der gegnerischen Aktionen nicht beachtet wird, in welcher Situation die entsprechende Aktion gemacht wurde.

3.4.2 Pokerbots für Online-Poker-Plattformen

Bei den Pokerbots, die auf echten Pokerplattformen spielen können, reicht das Spektrum von dubiosen Onlineangeboten, die damit locken viel Geld

mit dem Einsatz ihres Bots erspielen zu können, bis zu komplett freien und seriösen Softwareprojekten. Die zusätzliche Aufgabe, die diese Art von Bots erledigen muss, ist die Interaktion mit der Pokerplattform. Insbesondere das Auslesen des Spielzustands, ist ein nicht zu unterschätzendes Problem.

- *Open Holdem Bot* - Bei Open Holdem Bot (OH) handelt es sich um ein Framework für die Entwicklung eines Pokerbots. OH ist aus dem, zunächst kommerziell vertriebenen, WinHoldEm Bot entstanden, weil die Entwickler mit der Pflege der Software nicht hinterher kamen. OH arbeitet bildbasiert. Es werden in Echtzeit, Screenshots von dem Bildschirm gemacht und die Bildschirminformationen ausgelesen. Dazu muss der Benutzer ein sogenanntes Tablemap erstellen, indem er die Software OpenScrape dazu benutzt, semantische Annotationen zu der Benutzeroberfläche der Pokerplattform zu machen. In dem OH Framework ist bereits eine einfache, automatisch spielende KI enthalten. Diese ist allerdings sehr simpel und spielt nicht gewinnbringend. Dadurch, dass von verschiedenen Personen, über einen langen Zeitraum an dem Programm gearbeitet worden ist, haben sich einige Softwaretechnische Probleme gebildet. Programmteile sind teilweise sehr unsauber geschrieben und schlecht dokumentiert. Außerdem haben sich Online-Poker-Plattformen visuell weiterentwickelt. Um in dem OH Framework diesen Weiterentwicklungen zu begegnen, müssen sehr umständliche Anpassungen gemacht werden.
- *Holdem Poker Bot* [31] ist ein kommerzieller Pokerbot. Sein Quellcode ist nicht frei zugänglich. Dennoch lässt sich feststellen, dass der Bot die Informationen über den aktuellen Spielstand per DLL-Injection bezieht. Laut Bedienungsanleitung [30] funktioniert der Bot ausschließlich, wenn die Software der Pokerplattform so eingestellt ist, dass im Chatfenster alle Informationen eingetragen werden. Daher wird angenommen, dass alle Informationen per DLL-Injection aus diesem Fenster gelesen werden. Um das Ausnutzen dieser Sicherheitslücke zu vermeiden, haben viele Pokerplattformen diese Option bereits aus ihrer Software entfernt. Der Bot verwendet eine KI, die angeblich in einer Zeitspanne von zwei Jahren entwickelt wurde. Zusätzlich existiert eine intuitive Scriptsprache, um komplett eigene Strategien für den Bot zu entwerfen. Der Anbieter verspricht keine Gewinne durch die Anwendung der 129$ teuren Software.
- Torbjörn Lofterud stellte, im Rahmen des Chaos Communication Camp 2011, seine Erfahrungen [18] mit der Entwicklung von Pokerbots vor. Ein von ihm entwickelter Bot, arbeitet mit einer Informationsextraktion über die Analyse von Netzwerk-Traffic. Dabei wurde eine Schwachstelle bei der Zertifizierung, zwischen dem Client und Server der Pokerplattform, ausgenutzt. Dadurch ist es möglich viele Informationen in einem standardisierten Format auszulesen. Das größte Problem dabei ist,

dass nicht alle Informationen übertragen werden. So fehlt zum Beispiel die Information, welche Aktionen einem Spieler zur Verfügung stehen. Die Lösung von diesem Problem wurde in dem Beitrag verschwiegen. Ein weiterer Nachteil ist, dass die Schwachstelle in der Netzwerkverbindung jederzeit von den Plattformbetreibern geschlossen werden kann. Dann erhält der Bot keinerlei Informationen mehr. Dem Entwickler ist es, selbst nach langer Entwicklungszeit, nicht gelungen, eine gewinnbringende Strategie zu programmieren und er sieht sein Projekt daher als gescheitert an.

- Andere Pokerbots, sind zahlreich im Internet zu finden. Die Betreiber der Seiten verkaufen sie für geringe Geldbeträge oder bieten sie zum freien Download an. Es wird mit unrealistischen Versprechungen geworben. Tatsächlich handelt es sich bei solchen Angeboten um Blender oder Computerviren. Die große Zahl der verschiedenen Angebote zeigt das starke Interesse an Pokerbots.

Kapitel 4

Konzept der entwickelten Software

Das Ziel dieses Kapitels ist es die Funktionsweise, Architektur, sowie das Konzept der ,im Rahmen dieser Arbeit entwickelten Software (im Folgenden als *PokerBot* bezeichnet), vorzustellen. Dabei wird häufig Bezug auf die bisher behandelten Themen genommen. In Abschnitt 4.1 werden die Anforderungen an PokerBot diskutiert, bevor in Abschnitt 4.2, die einzelnen Module und in Abschnitt 4.3 die übrigen Bestandteile, der entwickelten Software, näher erläutert werden. Dabei wird auch auf die Schnittstellen und intermodularen Abläufe eingegangen. Die Vertiefung in einzelne Themen erfolgt anschließend in Kapitel 5.

4.1 Anforderungen an den PokerBot

Aus Kapitel 3 ist bekannt, dass bestimmte Pokerstrategien es ermöglichen, die eigenen Gewinnchancen positiv zu beeinflussen. Besonders bei schwachen Gegnern, erweist sich die Einhaltung relativ simpler Regeln als wirkungsvoll. Solche Strategien erfordern ein diszipliniertes Verhalten und bieten auf lange Sicht einen konstanten, aber niedrigen Gewinn. Damit der Gewinn für einen Online-Poker-Spieler lukrativ wird, muss sehr viel gespielt werden, mit hohen Einsätzen gespielt werden oder es wird von der disziplinierten Strategie abgewichen und auf schnellere Gewinne spekuliert. Das Spiel mit hohen Einsätzen ist, aufgrund stärkerer Gegner, anspruchsvoller als das Spiel mit niedrigen Einsätzen. Auf dieser Basis werden schnell die offensichtlichen Stärken eines computergesteuerten Spielers deutlich. Er kennt keine Langeweile, keine Müdigkeit und keine Risikobereitschaft.

4.1.1 Spielerische Anforderungen

Nach der Untersuchung verschiedener Pokerbots (vgl. Abs. 3.4) wurden für PokerBot folgende Anforderungen festgelegt:

- PokerBot soll über einen langen Zeitraum konstant und möglichst fehlerfrei eine risikoarme Strategie verfolgen.
- Das simultane Spielen vieler Partien ist wünschenswert. Es ermöglicht einen höheren Gewinn, bei niedrigen Einsätzen und minimiert zugleich die Zufallskomponente.
- Andere Spieler könnten möglicherweise einen Vorteil erhalten, wenn PokerBot sich als computergesteuerter Spieler zu erkennen gibt, weshalb unbedingt darauf zu achten ist, dass PokerBot sich wie ein menschlicher Spieler verhält. Merkmale, wie eine variable Reaktionszeit und das Unterlassen unrealistischer, simultaner Aktionen müssen beachtet werden. Um auszuschließen, dass für PokerBot spielerische Nachteile entstehen, soll auch den Betreibern der Pokerplattformen nicht erkenntlich sein, dass es sich um einen computergesteuerten Spieler handelt.
- Die Unterstützung unterschiedlicher Pokerplattformen, ermöglicht es 24 Stunden am Tag zu spielen, ohne Verdacht zu erregen und ist daher wünschenswert.
- Risikoarme Pokerstrategien sind auf Partien mit möglichst vielen Spielern ausgelegt. PokerBot muss demnach in der Lage sein, einen ungünstigen Tisch zu erkennen und zu verlassen.
- Manche Pokerstrategien greifen auf zusätzliche Informationsquellen zu, um sich einen weiteren Vorteil gegenüber den Mitspielern zu verschaffen. Denkbar ist beispielsweise die Klassifizierung der Gegner.

4.1.2 Technische Anforderungen

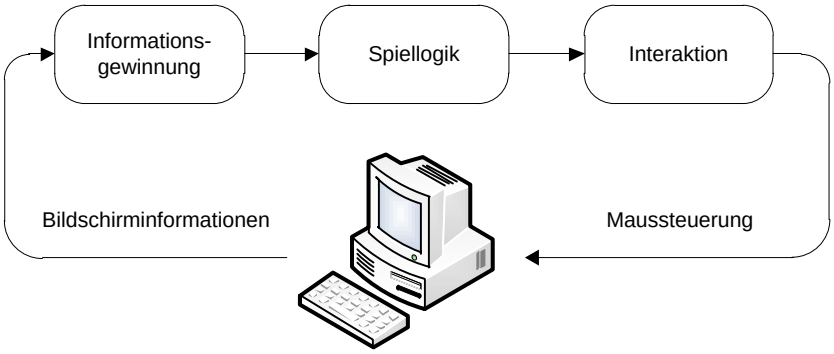
Damit ein Expertensystem erfolgreich Poker spielen kann, sind aus technischer Sicht, drei Aufgaben zu bewältigen:

- Der aktuelle Spielzustand muss erkannt werden.
- Aus den Informationen über den Spielzustand muss entschieden werden, welche Aktion ausgeführt werden soll.
- Anschließend muss über die grafische Benutzeroberfläche die Aktion ausgeführt werden.

4.2 Module

Aus den technischen Anforderungen, lässt sich ein System mit drei Modulen ableiten: (1) Informationsgewinnung, (2) Spiellogik und (3) Interaktion (siehe Abb. 4.1). Jedes Modul bewerkstelligt seine Aufgabe autonom und

Abbildung 4.1: Grundmodule



kommuniziert, über eine klar definierte Schnittstelle, mit den anderen Modulen. Trotz dieser einfachen und nur groben Unterteilung, lässt sich bereits der zentrale Teil des Expertensystems, nämlich die Spiellogik, völlig unabhängig jeglicher äußeren Einwirkungen, wie insbesondere der Wahl der Pokerplattform, entwickeln.

4.2.1 Modul 1: Informationsgewinnung

Dieses Modul ist dafür zuständig, die Bildschirminformationen aufzunehmen und soweit zu verarbeiten, dass die Spiellogik sie interpretieren kann.

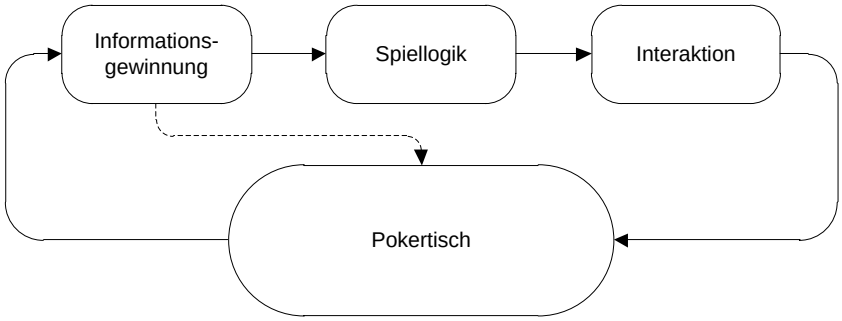
- In einem ersten Schritt werden alle angeschlossenen Bildschirme nach sichtbaren Tischen durchsucht. Position, Größe und Art der gefundenen Tische werden abgespeichert.
- Im nächsten Schritt wird ein gefundener Pokertisch, von der Informationsgewinnung, solange beobachtet, bis festgestellt wird, dass PokerBot am Zug ist.
- Anschließend wird, im dritten Schritt, ein Bild dieses Pokertisches erzeugt und als Grundlage für die weitere Verarbeitung verwendet. Aus dem Bild wird der aktuelle Spielzustand ausgelesen und schließlich an die Spiellogik weitergeleitet.

Während der erste Schritt bei Programmstart manuell, über eine GUI, ausgeführt wird, sind die Schritte zwei und drei, fest in den automatischen Programmablauf integriert. Diese werden in Abbildung 4.2 nochmals veranschaulicht.

Simultanes Spielen

Der beschriebene Ablauf lässt sich auf das simultane Spielen an mehreren Tischen übertragen. Alle im ersten Schritt gefundenen Tische werden gleich-

Abbildung 4.2: Programmablauf nach der Initialisierung



zeitig beobachtet, bis PokerBot an einem Tisch an der Reihe ist. Von diesem Tisch wird ein Bild aufgenommen, weiterverarbeitet und an das nächste Modul weitergeleitet. Erst wenn von der Interaktion eine Aktion ausgeführt wurde, setzt das Informationsgewinnungsmodul die Beobachtung aller Tische fort. Auf diese Weise werden alle simultan gespielten Spiele nach einander behandelt. Diese Vorgehensweise ist notwendig, da alle Eingaben von nur einer Maus eingegeben werden sollen (siehe Abs. 4.2.4). Abbildung 4.3 zeigt den Ablauf mit mehreren Tischen.

4.2.2 Modell des aktuellen Spielzustands

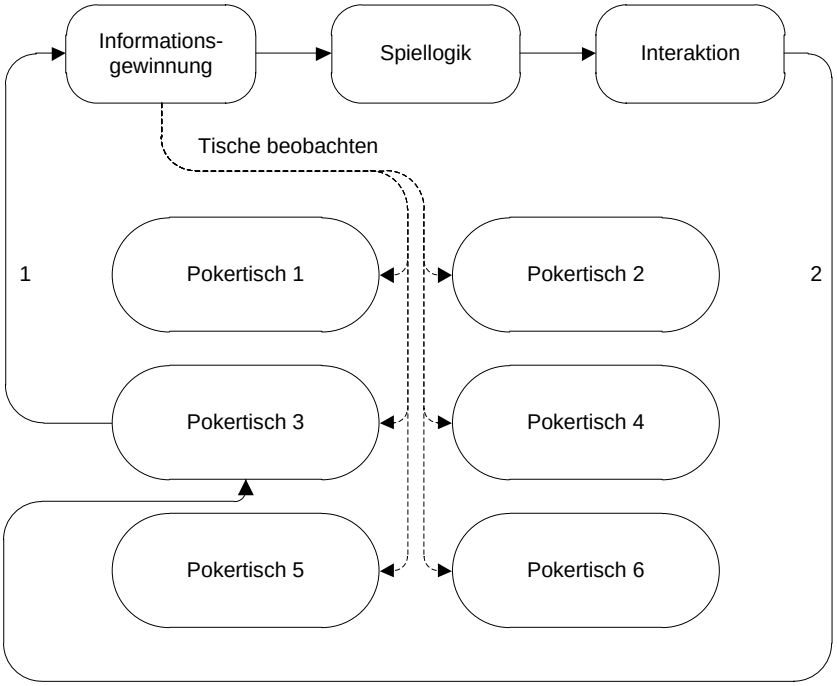
Die Schnittstelle zwischen der Informationsgewinnung und der Spiellogik lässt sich als vollständiges Modell des Spielzustands beschreiben. Zur Veranschaulichung kann ein Modell als eine Tabelle verstanden werden. Für jedes neue Spiel wird eine neue Tabelle erzeugt, in der jede Wettrunde eine eigene Spalte einnimmt(vgl. Tabelle 4.1). Ein Modell beinhaltet alle für die Spiellogik relevanten Informationen und einige zusätzliche Metadaten[1]. Es wird genau dann aktualisiert, wenn PokerBot am Zug ist und eine Aktion ausgeführt werden soll. Dabei wird eine neue Spalte mit allen Ereignissen seit der letzten Aktion am Ende der Tabelle eingefügt. Das Erhalten aller Datensätze und Modelle ermöglicht der Spiellogik frei zu entscheiden, welche Daten später für die Entscheidungsfindung herangezogen werden. Es ist dadurch außerdem möglich, komplette Spielsituationen zu simulieren oder mit unterschiedlichen Parametern zu wiederholen.

4.2.3 Modul 2: Spiellogik

Die aus dem ersten Modul kommenden Daten in Form des Modells des aktuellen Spielzustands werden in dem Spiellogikmodul benutzt, um zu entscheiden, welche Aktion von PokerBot gemacht werden soll. Als Eingabe erhält es

[1] Datum, Uhrzeit, Pokerplattform

Abbildung 4.3: Programmablauf bei mehreren Pokertischen. Sobald PokerBot an einem Tisch am Zug ist (Pokertisch 3) und das erste Bild eines Tisches erzeugt wird (Pfeil 1), hört die Informationsgewinnung auf die Tische zu beobachten (gestrichelter Pfeil), bis vom Interaktionsmodul eine Aktion ausgeführt wurde (Pfeil 2).



das Modell und liefert, als Ausgabe, eine Aktion. Der Ablauf des zweiten Moduls kann in einzelne Schritte aufgeteilt werden. Abbildung 4.4 visualisiert die Schritte des Spiellogikmoduls.

Natürlichsprachliches Regelsystem als Kern KI

Der Kern der Künstlichen Intelligenz von PokerBot basiert auf einem Regelsystem, nach dem menschliche Spieler sehr langsam, aber konstant, Gewinne erzielen sollen [25]. Dieses Regelsystem beschreibt genau in welcher Situation welche Aktion gemacht werden soll. Wir haben uns für diese Strategie entschieden, weil die Herausgeber der Strategie jedem Spieler, der sie anwendet 50$ Startkapital überweisen, um anschließend an den Gewinnen der Spieler beteiligt zu werden. Die Strategie hat bei einem Testversuch funktioniert, war aber sehr langwierig und langweilig, weshalb ein langes Spielen nach dem Regelsystem, nicht durchgehalten wurde.

Um letztendlich die, nach dem Regelsystem, richtige Aktion zu ermitteln,

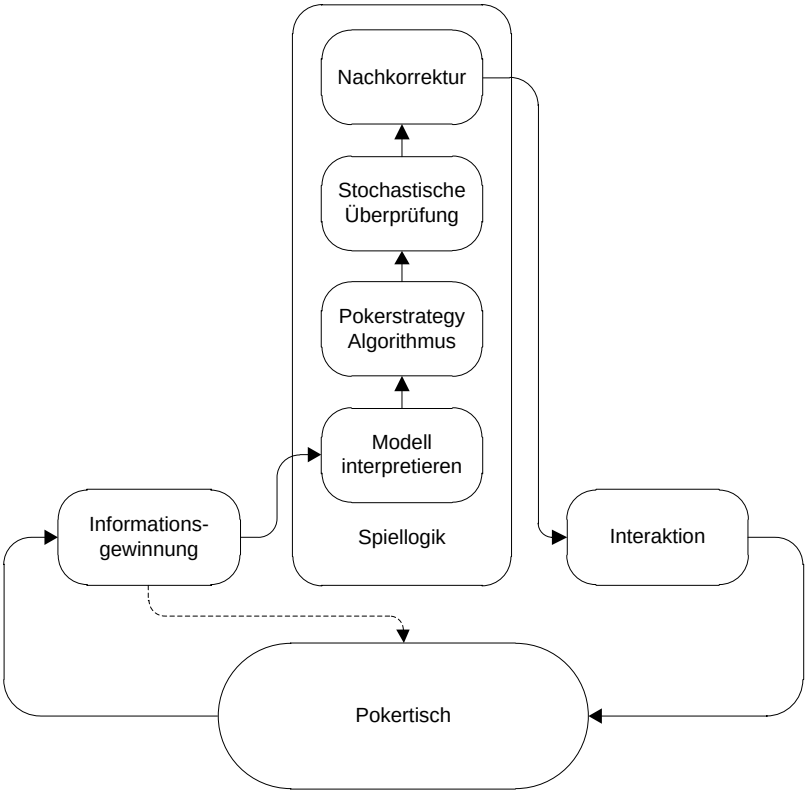
Tabelle 4.1: Beispiel für ein Modell des Spielzustands

Datum	09.11.2011	09.11.2011	09.11.2011
Uhrzeit	02:35	02:36	02:36
Plattform	MusterRaum	MusterRaum	MusterRaum
Limit	0.10	0.10	0.10
SB-Höhe	0.05	0.05	0.05
Eigene-Position	5	5	5
Dealer-Position	1	1	1
SB-Position	2	2	2
BB-Position	3	3	3
Spieleranzahl	8	8	8
Aktive Spieler	8	6	3
Spiel	1	1	1
Hand	1	1	1
Phase	Preflop	Flop	Flop
Runde	1	1	2
Handkarten	A♣, A♢	A♣, A♢	A♣, A♢
Gem.-Karten	-	A♠, A♡, 7♢	A♠, A♡, 7♢
Pot	0.25	1.05	1.35
S1-Name	Whale	Whale	Whale
S1-Aktion	Call	Raise	Raise
S1-Stack	34.81	34.71	34.61
S1-Aktiv	true	true	true
S2-Name	Donk	Donk	Donk
S2-Aktion	Call	Call	Fold
S2-Stack	4.05	3.95	3.95
S2-Aktiv	true	true	false
...	...	...	...

muss zunächst geklärt werden, welche Situation genau vorliegt. Dazu sind folgende Fragen zu klären:

- In welcher Phase befindet sich das Spiel gerade? (Preflop, Flop, River, Turn)
- Was habe ich für Karten?
- Was habe ich für ein Blatt? (Highcard, Paar, Straße, Flush, Draw, ...)
- An welcher Position bin ich? (Früh, Mittel, Spät, Small Blind, Big Blind, Dealer)
- Was haben die anderen Spieler bis jetzt gemacht? (Ein Gegner hat erhöht und mindestens einer ist mitgegangen. Mehr als zwei Gegner haben erhöht. Alle haben geschoben. ...)
- Wie viel Geld liegt bereits im Pot?

Abbildung 4.4: Modul 2: Spiellogik im Programmablauf



Nachdem die Situation mit Hilfe der eingeklammerten Antworten formalisiert ist, kann der Spieler im Regelsystem nachschauen, welche Aktion er ausführen soll. Zur Implementierung dieses Regelsystems in PokerBot sind folgende zwei Schritte durchzuführen:

- 1. Die Aufbereitung der Daten aus dem Model des aktuellen Spielzustands 5.3.1. Dazu gehört das Ermitteln des eigenen Blattes, das Einteilen der eigenen Position in die genannten Klassen, sowie die Interpretation der Aktionen der Mitspieler.
- 2. Die Formalisierung des natürlichsprachlichen Regelsystems 5.3.2.

Stochastische Überprüfung

Während der Implementierung der KI hat sich herausgestellt, dass die natürlichsprachliche Strategie sich nicht gänzlich formalisieren lässt(siehe Abs.5.3.2). Um die aus Sonderfällen resultierenden Fehlentscheidungen abzufangen, überprüft der nächste Schritt im Spiellogikmodul anhand stochastischer Berechnungen 5.3.3 die zuvor getroffene Entscheidung und korrigiert diese unter

Umständen. Dieser Schritt ist optional und kann über die grafische Benutzeroberfläche (Abb. 4.5) ein- und ausgeschaltet werden (vergl. Abs. 4.3.1).

Nachkorrektur

In einem letzten Schritt wird diese Entscheidung nochmals überprüft, da es bestimmte Situationen gibt, in denen die vorherigen Schritte vorschlagen, aufgrund beispielsweise sehr schlechter Karten, auszusteigen. Hat kein anderer Spieler in der aktuellen Wettrunde eine Erhöhung gemacht, dann ist es immer besser zu Schieben. Ist ein *kostenloses* Schieben möglich, wird die Entscheidung durch diesen Schritt dahingehend korrigiert. Eine Korrektur wird auch durchgeführt, wenn vorgeschlagen wurde zu erhöhen, obwohl die maximale Anzahl an Erhöhungen schon erreicht wurde.

4.2.4 Modul 3: Interaktion

Das dritte Modul ist das kleinste Modul. Es bekommt mitgeteilt an welchem Tisch, welche Aktion durchgeführt werden soll und geht dann folgende Schritte durch:

- Auslesen der Koordinaten, des Tisches.
- Berechnen der Position, an der sich die richtige Schaltfläche befindet.
- Mauszeiger zu dieser Position bewegen und klicken.

Dabei wird darauf geachtet, dass weder die Bewegungsgeschwindigkeit des Mauszeigers, noch die genauen Positionen, auf die geklickt wird, auf einen computergesteuerten Spieler schließen lassen. Künstliche Reaktionsverzögerungen und andere Zufallsparameter sind in die Mausbewegungen eingebaut.

4.3 Weitere Bestandteile von PokerBot

Neben den drei Hauptmodulen, besteht die, im Rahmen dieser Arbeit, entwickelte Software, aus einigen anderen Komponenten. Die Wichtigsten dieser Bestandteile werden in den folgenden Abschnitten vorgestellt.

4.3.1 Grafische Benutzeroberfläche

Über die grafische Benutzeroberfläche (Abb. 4.5) wird vom Benutzer, die Initialisierung von PokerBot vorgenommen und die Überwachung durchgeführt. Das Statusfenster zeigt dem Benutzer, ausgewählte Informationen. Es können außerdem Spielverläufe gespeichert und vergangene Spiele, zu Analysezwecken, geladen werden. Über den Reiter Bot Konfigurator wird die Ansicht 4.6 erreicht. Hier können die wichtigsten Variablen der mathematischen Überprüfung beeinflusst oder auch komplett deaktiviert werden. Die Variablen lassen sich im laufenden Betrieb ändern. Zusätzlich lassen sich die Analysewerkzeuge *Tabelle* und *OnScreen Display* ein- und ausschalten.

Abbildung 4.5: Grafische Benutzeroberfläche von PokerBot

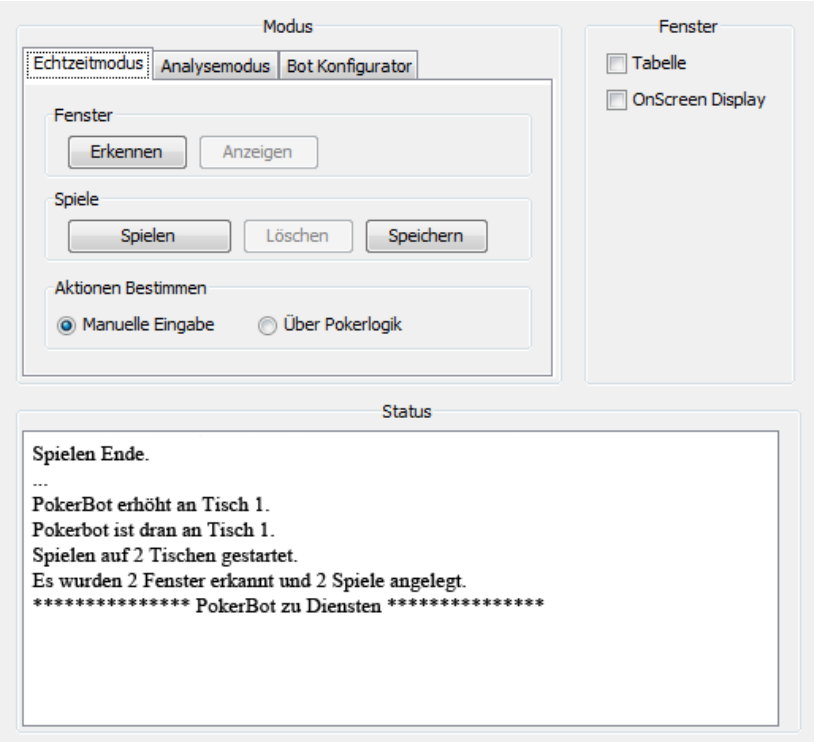
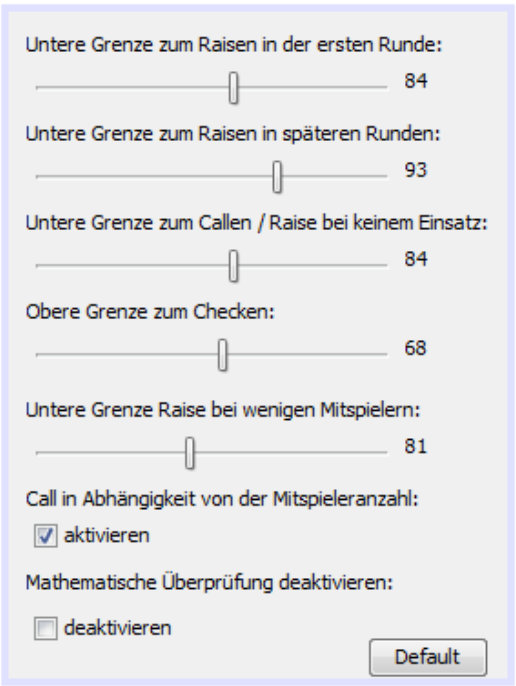


Abbildung 4.6: Einstellungsmöglichkeiten für die KI



4.3.2 Test- und Analysewerkzeuge

Für die Weiterentwicklung und Wartung der Software, sowie für das Testen neuer Algorithmen für die Spiellogik, stehen einige Werkzeuge bereit.

OnScreen Display

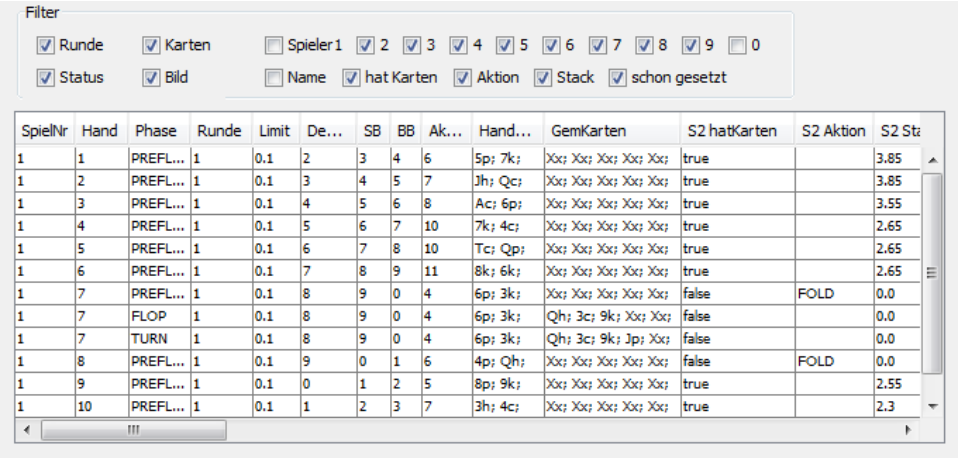
Das Einblenden des OnScreen Display's ermöglicht das schnelle Überprüfen der Informationsgewinnung. Die blauen und roten Kästen auf Abbildung 4.7 zeigen das OnScreen Display. Rote Kästen zeigen die Informationen der Spieler auf den beiden Blind-Positionen, sowie Informationen zum Dealer. Die blauen Kästen beinhalten die Daten der übrigen Spieler. Es werden vier Informationen aus der aktuellen und der vorherigen Wettrunde angezeigt: Der *toCall* Wert, die getätigte Aktion, der Stack des Spielers und der Name des Spielers. Was auf dem OnScreen Display angezeigt werden soll, ist im Programmcode leicht anpassbar. Die kleinen Schaltflächen auf den einzelnen Kästen ermöglichen es, deren Positionen zu speichern oder einzelne Kästen auszublenden.

Abbildung 4.7: Das Analysewerkzeug: OnScreen Display

Tabelle

Ein weiteres Werkzeug zur Analyse ist die Tabelle (siehe Abb. 4.8). Alle, im Modell des aktuellen Spielzustands enthaltenen, Informationen werden über die Tabelle visualisiert. Durch verschiedene Filter lässt sich die Tabelle an die aktuellen Bedürfnisse anpassen. Neben den Daten aus dem Modell wird außerdem ein Screenshot zu jeder Zeile der Tabelle gespeichert. Ein Klick auf die jeweilige Tabellenspalte zeigt den dazu gehörigen Screenshot an. Diese Funktionalität ermöglicht es, nachträglich, eine große Zahl an Händen und Spielen, bezüglich der Qualität der Informationsgewinnung, zu evaluieren. In Verbindung mit dem externen Werkzeug *Pokertracker* können, dank der Tabelle, gezielt und effizient, aufschlussreiche Erkenntnisse gewonnen werden.

Abbildung 4.8: Das Analysewerkzeug: Tabelle

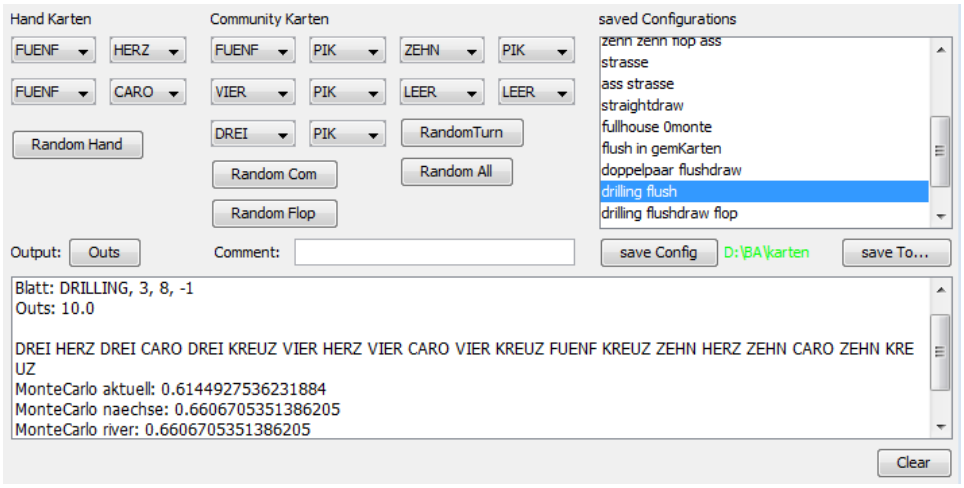


SpielNr	Hand	Phase	Runde	Limit	De...	SB	BB	Ak...	Hand...	GemKarten	S2 hatKarten	S2 Aktion	S2 Sta
1	1	PREFL...	1	0.1	2	3	4	6	5p; 7k;	Xx; Xx; Xx; Xx; Xx;	true		3.85
1	2	PREFL...	1	0.1	3	4	5	7	Jh; Qc;	Xx; Xx; Xx; Xx; Xx;	true		3.85
1	3	PREFL...	1	0.1	4	5	6	8	Ac; 6p;	Xx; Xx; Xx; Xx; Xx;	true		3.55
1	4	PREFL...	1	0.1	5	6	7	10	7k; 4c;	Xx; Xx; Xx; Xx; Xx;	true		2.65
1	5	PREFL...	1	0.1	6	7	8	10	Tc; Qp;	Xx; Xx; Xx; Xx; Xx;	true		2.65
1	6	PREFL...	1	0.1	7	8	9	11	8k; 6k;	Xx; Xx; Xx; Xx; Xx;	true		2.65
1	7	PREFL...	1	0.1	8	9	0	4	6p; 3k;	Xx; Xx; Xx; Xx; Xx;	false	FOLD	0.0
1	7	FLOP	1	0.1	8	9	0	4	6p; 3k;	Qh; 3c; 9k; Xx; Xx;	false		0.0
1	7	TURN	1	0.1	8	9	0	4	6p; 3k;	Qh; 3c; 9k; Jp; Xx;	false		0.0
1	8	PREFL...	1	0.1	9	0	1	6	4p; Qh;	Xx; Xx; Xx; Xx; Xx;	false	FOLD	0.0
1	9	PREFL...	1	0.1	0	1	2	5	8p; 9k;	Xx; Xx; Xx; Xx; Xx;	true		2.55
1	10	PREFL...	1	0.1	1	2	3	7	3h; 4c;	Xx; Xx; Xx; Xx; Xx;	true		2.3

Blattsimulator

Ein weiteres Hilfsprogramm wurde für das schnelle Testen von mathematischen Berechnungen entwickelt. Die Oberfläche ist auf Abbildung 4.9 zu sehen. Das Programm greift direkt auf den Programmcode von PokerBot zu und verwendet genau die Methoden, welche der Bot für die Entscheidungsfindung benutzt. Über Drop-Down Menüs werden die Handkarten und Gemeinschaftskarten konfiguriert. Durch einen Klick auf den Knopf Outs werden die Gewinnwahrscheinlichkeiten und Outs errechnet und anschließend im Ausgabefeld angezeigt. Die eingegeben Handkarten und Gemeinschaftskarten lassen sich in einer Datei speichern und laden. Situationen können schneller getestet werden, da die Karten nicht nach jedem Neustart neu eingegeben werden müssen.

Abbildung 4.9: Das Hilfsprogramm zum Testen der mathematischen Entscheidungsfindung



Kapitel 5

Implementierung

In diesem Abschnitt der Arbeit werden wesentliche Teile der Implementierung herausgearbeitet. In Abschnitt 5.1 werden Bereiche vorgestellt, die die gesamte Architektur betreffen. Anschließend werden, in den Abschnitten 5.2 bis 5.4, die drei Module der Software ausführlich behandelt. Oftmals werden konkrete Programmausschnitte, als Quelltext, zu den Erklärungen hinzugezogen. Diese stammen, aus dem im Rahmen dieser Arbeit entwickelten Programm, sind aber unter Umständen abgewandelt. Der vollständige Programmcode ist unter [14] einsehbar. Die verwendete Programmiersprache ist Java.

5.1 Modulübergreifende Programmteile

Wichtige Datenstrukturen und Programmteile, welche die Kommunikation zwischen den einzelnen Modulen sicherstellen, werden in diesem Abschnitt behandelt. Außerdem werden die Initialisierungsroutine und die Hauptschleife im Detail vorgestellt.

5.1.1 Unterstützung verschiedener Pokerplattformen

Ein grundsätzliches Problem, bei der Entwicklung von Pokerbot, war die Unterstützung mehrerer Pokerplattformen. Aus diesem Grund ist jegliche Funktionalität, die in direktem Zusammenhang mit einer externen Pokersoftware steht, gekapselt. Für jede unterstützte Pokerplattform muss das Interface *Pokerplattform* implementiert werden. Das Interface besteht, zum größten Teil, aus Methoden, welche Auskunft über das Erscheinungsbild des Pokerraums geben. Beispielsweise, über die Rückgabe von Größen oder Positionen einzelner Elemente. Der Quelltext 5.1 zeigt hierzu einige Beispiele.

Quelltext 5.1: Beispiele aus dem Interface Pokerplattform

```
1 public interface Pokerplattform {
2   public Plattform getPlattform();
```

```
3   public Koordinate getKartenPositionen(int index);
4   ...
5 }
6 public class BeispielPokerPlattform implements Pokerplattform {
7   public Plattform getPlattform() {
8     return Plattform.BeispielPokerPlattform;
9   }
10  public Koordinate getKartenPositionen(int index) {
11    Koordinate[] karte = new Koordinate[7];
12    index = (index < 0) ? 0 : index;
13    index = (index > karte.length) ? karte.length : index;
14    karte[0] = new Koordinate(256, 11);
15    karte[1] = new Koordinate(270, 15);
16    ...
17    return karte[index];
18  }
19 ...
20 }
```

Weitere zu implementierende Methoden liefern als Rückgabewert Templates[1] oder Alphabete[2]. Beispiele hierzu zeigt der Quelltext 5.2.

Quelltext 5.2: weitere Beispiele aus dem interface Pokerplattform

```
1 public interface Pokerplattform {
2   public BufferedImage getLogo();
3   public Alphabet getZahlenAlphabet();
4   ...
5 }
6
7 public class BeispielPokerPlattform implements Pokerplattform {
8
9   private Alphabet zahlenAlphabet = bildverarbeitung.Texterkennung.
      ladeAlphabetAusOrdner("img/BeispielPokerPlattform/zeichen/",
      Texterkennung.AlphabetTyp.nummerisch);
10
11    public Alphabet getZahlenAlphabet() {
12        return this.zahlenAlphabet;
13    }
14
15    private BufferedImage logo = bildverarbeitung.BV.loadImageFromFile("
      img/BeispielPokerPlattform/logo.png");
16
17    public BufferedImage getLogo() {
18        return this.logo;
19    }
20 ...
21 }
```

In seltenen Fällen ist die Implementierung etwas komplexerer Methoden notwendig. Das ist erforderlich, wenn die Unterschiede der Pokerplattformen sich

[1] Bildausschnitte, welche von der Bildverarbeitung bei der Informationsgewinnung (siehe Abs. 5.2.1) verwendet werden, um Bildbereiche wiederzuerkennen oder zu klassifizieren.

[2] Listen aus Paaren von je einem Template und einer Beschreibung dieses Templates.

nicht auf rein optische Merkmale reduzieren lassen. Beispielsweise ist die in Quelltext 5.3 zu sehende Methode ebenfalls Bestandteil des vorgestellten Interfaces. Diese Methode wird aufgerufen, wenn überprüft wird, ob PokerBot am Zug ist. Auf den meisten Pokerplattformen kann am Vorhandensein der *Fold-Schaltfläche* erkannt werden, dass PokerBot am Zug ist. Es gibt allerdings auch Ausnahmen für die anders erkannt werden muss, ob PokerBot an der Reihe ist. Das Interface Pokerplattform bietet die Freiheit, solche Besonderheiten korrekt zu behandeln.

Quelltext 5.3: Die Methode `guckObDran()` aus dem Interface Pokerplattform.

```
1
2     public boolean guckObDran(Fenster fenster, BufferedImage screenshot)
       {
3         BufferedImage fensterSubScreenshot = fenster.getSubScreenshot(
      screenshot);
4         BufferedImage buttonTemplate = BV.loadImageFromFile("img/
      BeispielPokerPlattform/foldButton.png");
5         BufferedImage buttonBereich = fensterSubScreenshot.getSubimage
      (390, 486, 10, 20);
6         if (BV.templateMatching(buttonBereich, buttonTemplate, 30).size
      () > 0) {
7             return true;
8         }
9         return false;
10    }
```

5.1.2 Datenstrukturen

Um die Hauptschleife und die Initialisierungsmechanismen besser erläutern zu können, werden die wichtigsten Datenstrukturen vorgestellt. Die Zentrale Datenstruktur ist eine `ArrayList<Spiel>`. Die Elemente dieser Liste entsprechen den tatsächlichen Spielen an denen PokerBot teilnehmen kann. Die Klasse *Spiel* dient der Synchronisation der Bildschirminformationen mit den Spielinformationen. Sie hat eine Variable vom Typ `Fenster` und eine vom Typ `ArrayList<Runde>`. In der Klasse `Fenster` werden beispielsweise Koordinaten, Abmessungen und die `Pokerplattform` des jeweiligen Spiels gespeichert, während die `ArrayList<Runde>` das in Abschnitt 4.2.2 vorgestellte Modell realisiert. Der Inhalt der Klasse `Runde` kann aus dem Quelltext 5.4 entnommen werden. Jede `Runde` entspricht einer Spalte der Tabelle 4.1.

Quelltext 5.4: Datenstruktur zur Speicherung des Modells

```
1  public class Runde {
2
3      //allgemeine Spielinformationen
4      public Plattform plattform;
5      public double limit;
6      public double sbBet;
7
8      //sichtbare Karten
9      public Karte[] handKarten;
10     public Karte[] gemKarten;
11
12     //Positionen
13     public int meinePos;
14     public int dealerPos;   //DealerButton
15     public int sbPos;       //SmallBlind
16     public int bbPos;       //BigBlind
17
18     //Spielverlauf
19     public int spiel;
20     public int hand;
21     public Phase phase;
22     public int runde;
23
24     public double pot;
25     public int aktiveSpieler;
26     public int spielerAmTisch;
27
28     //genauere Informationen zu allen Gegnern
29     public Spieler[] spieler;
30
31     //sichtbare Buttons zum Zeitpunkt der Aktualisierung
32     public Buttons[] verfuegbareButtons;
33
34 }
35
36 public class Spieler {
37
38     private double schonGesetzt;
39     private Aktion Aktion;
40     private double stack;
41     private String name;
42     private boolean hatKarten;
43     private boolean istDa;
44
45     //es folgen getter und setter
46     ...
47 }
```

Enumerationen und Hilfsklassen

An einigen Stellen werden im Quelltext Enumerationen und kleinere Hilfsklassen verwendet. Das ist notwendig, um den spezifischen Anforderungen der Domäne Poker gerecht zu werden. In diesem Abschnitt werden die wichtigsten Hilfsklassen genannt, damit die folgenden Abschnitte besser verständlich sind. Neben den in Abschnitt 5.1.2 genannten Klassen Spiel und Runde, gibt es noch die Hilfsklassen Spieler und Karte. Die Klasse Spieler ist im bereits erwähnten Quelltext 5.4 aufgeführt und ist, wie `Runde`, zur Datenspeicherung vorgesehen. Die Klasse `Karte` hat zwei Variablen und bietet die Möglichkeiten diese Variablen auf verschiedene Weise abzufragen. Die Variablen entsprechen den beiden Eigenschaften Kartenwert und Kartenfarbe von echten Spielkarten. Viele in dem Programm verwendete Datentypen sind durch Java-Enumerationen (`Enum`) definiert. Alle Enums sind angelehnt an Pokerbegriffe und verbessern die Übersicht im Quelltext erheblich. Die folgende Auflistung bietet eine Übersicht der, am häufigsten verwendeten, Enums:

- Eigenschaften von Karten
  - `Wert`: ZWEI, DREI, VIER, ..., DAME, KOENIG, ASS, LEER
  - `Farbe`: HERZ, CARO, PIK, KREUZ, LEER
- Mögliche Aktionen der Spieler
  - `Aktion`: RAISE, CHECK, CALL, FOLD, KEINE
- Phasen in die ein Pokerspiel unterteilt ist
  - `Phase`: PREFLOP, FLOP, TURN, RIVER
- Alle unterstützten Plattformen
  - `Pokerplattform`: BEISPIELPOKER, MUSTERPOKER, ...
- Eigenschaften für die Pokerlogik
  - `Blatt`: NICHTS, PAAR, ZWEIPAAR, DRILLING, OESD,...
  - `PreFlopBlatt`: SEHRSTARK, STARK, SPEKULATIV,...
  - `GegnerAktion`: ALLERAUS, EINERMIT, ALLECHECK, EINERHOCH, ZWEIODERMEHRMIT...

5.1.3 Initialisierungsroutine

Nach dem Start der Software ist es möglich, PokerBot selbstständig online spielen zu lassen. Vor dem eigentlichen Spielstart muss allerdings noch eine Initialisierungsroutine ablaufen. Alle Tische, an denen PokerBot spielen soll, werden, von dem Benutzer, sichtbar auf den angeschlossenen Bildschirmen platziert. Anschließend wird die *Erkennen*-Schaltfläche betätigt, um die Initialisierung zu starten. Dabei wird in einer Schleife über alle angeschlossenen

Bildschirme jeweils ein Screenshot gemacht, in ein Schwarz-Weiß-Bild konvertiert und die Methode `ArrayList<Fenster> getSichtbareFenster(...)`[3] ausgeführt. Für jedes so gefundene `Fenster` wird ein neues `Spiel` angelegt und das `Fenster` an den Konstruktor übergeben. Um den Initialisierungsvorgang abzuschließen werden die neu erzeugten Spiele an die Klasse PokerBot übergeben. Dort werden sie in einer *Membervariablen* gespeichert. Der Quelltext 5.5 zeigt die Methode, welche für die Initialisierung zuständig ist.

Quelltext 5.5: Diese Ausschnitt zeigt die Methode die zum Initialisieren aufgerufen wird.

```
1  private void buttonErkennenActionPerformed(java.awt.event.ActionEvent
       evt) {
2
3    GraphicsEnvironment ge = GraphicsEnvironment.
       getLocalGraphicsEnvironment();
4    GraphicsDevice[] gs = ge.getScreenDevices();
5    ArrayList<Fenster> fensterListe = new ArrayList<Fenster>();
6    for (int i = 0; i < gs.length; i++) {
7      BufferedImage screenShot = BV.screenShot(i);
8      BufferedImage bwScreenShot = BV.rgbToGrey(screenShot);
9      ArrayList<Fenster> neueListe = Fenstererkennung.getSichtbareFenster(
       bwScreenShot, bot.getPokerPlattformen(), i);
10     for (Fenster fenster : neueListe) {
11       fensterListe.add(fenster);
12     }
13   }
14   ArrayList<Spiel> neueSpiele = new ArrayList<Spiel>();
15   for (int i = 0; i < fensterListe.size(); i++) {
16     neueSpiele.add(new Spiel(fensterListe.get(i)));
17   }
18   bot.setSpiele(neueSpiele);
19   this.textAreaStatus.insert("Es wurden " + fensterListe.size() + "
       Fenster erkannt und Spiele angelegt.\n", 0);
20 }
```

5.1.4 Hauptschleife

In der Klasse `PokerBot` befindet sich die Hauptschleife des Programms. Die Hauptschleife läuft in einem eigenen *Thread*, damit die grafische Benutzeroberfläche weiter ansprechbar bleibt. Dazu implementiert die Klasse `PokerBot` das Interface `Runnable`. Durch einen Klick auf die *Spielen*-Schaltfläche wird unter Anderem die Programmzeile

```
1 new Thread(bot).start();
```

ausgeführt. Das Interface `Runnable` sorgt dafür, dass dadurch die `run()`-Methode gestartet wird, welche die Hauptschleife beinhaltet. Diese wird im Folgenden schrittweise gezeigt und erläutert:

[3] In Abschnitt 5.2.3 wird ausführlich auf die Methode `ArrayList<Fenster> getSichtbareFenster(...)` eingegangen.

```
1 public void run() {
2   while (isRunning) {
3   try {
4   Thread.sleep(200);
5   GraphicsEnvironment ge = GraphicsEnvironment.
      getLocalGraphicsEnvironment();
6   GraphicsDevice[] gs = ge.getScreenDevices();
7
8   for (int screenID = 0; screenID < gs.length; screenID++) {
9     BufferedImage bwScreen = BV.screenShot(screenID);
10    bwScreen = BV.rgbToGrey(bwScreen);
```

In Zeile 4 wird sichergestellt, dass nur maximal fünf Schleifendurchgänge pro Sekunde stattfinden, um die Rechnerressourcen zu schonen. Diese Geschwindigkeit reicht aus, da PokerBot immer einige Sekunden Zeit hat, um einen Zug auszuführen[4]. In den Zeilen 5-10 werden -wie beim Initialisierungsprozess- Screenshots aller angeschlossenen Bildschirme gemacht und in Schwarzweiß-Bilder konvertiert. Dabei werden die Bildschirme nacheinander abgearbeitet.

```
10      for (Spiel spiel : spiele) {
11        if (screenID == spiel.fenster.screenID) {
```

Die Schleife in Zeile 10 iteriert über alle Spiele, welche durch den Initialisierungsprozess angelegt wurden. Alle Spiele, dessen Fenster sich auf dem aktuell abzuarbeitenden Bildschirm befinden (Zeile 11), werden durch die in Abbildung 4.3 dargestellten Module bearbeitet.

```
12          if (Informationsgewinnung.guckObDran(spiel.fenster, bwScreen))
       {
13            BufferedImage fensterBild = spiel.fenster.getSubScreenshot(
      bwScreen);
14            Informationsgewinnung.aktualisiereModell(spiel, fensterBild)
      ;
```

Das erste Modul prüft in Zeile 12, ob Pokerbot am Zug ist und -falls ja- wird ein Bild dieses Tisches erstellt(Zeile 13). Durch die nächste Zeile wird das Modell aktualisiert.

```
15            Aktion aktion = Aktionsbestimmung.aktionsBestimmung(spiel,
      ...);
16
17            Koordinate mausKoordinate = Maus.getMausKoordinate(spiel,
      aktion, ...);
18            gui.maus.mouseAktion(mausKoordinate, 1, screenID);
19            Interaktion.setEigeneAktion(spiel, aktion, bwScreen);
```

In Zeile 15 wird das aktualisierte Modell an eine Methode des zweiten Moduls übergeben, welche eine Aktion zurückgibt. Durch die Zeilen 16-18 wird diese Aktion von der Maus ausgeführt und die getätigte Aktion im Modell eingetragen. Aus Gründen der Übersicht wurden in den aufgeführten Programmausschnitten der Hauptschleife einige Dinge ausgelassen. Neben dem

[4] In der Regel zehn bis 30 Sekunden.

beschriebenem Ablauf werden in der Hauptschleife die Modelle auf die Festplatte gespeichert und die Analysewerkzeuge *OnScreen Display* und *Tabelle* aktualisiert. Außerdem finden einige zusätzliche Überprüfungen statt. Der vollständige Programmquelltext ist auf sourceforge [14] einsehbar.

5.2 Modul 1: Informationsgewinnung

In diesem Abschnitt wird die Implementierung des in Abschnitt 4.2.1 vorgestellten Konzepts behandelt. Dazu wird zunächst das Template Matching eingeführt, welches bei den meisten Bestandteilen des Informationsgewinnungsmoduls zum Einsatz kommt. Anschließend werden chronologisch einzelne Teile des Moduls besprochen. Die Fenstererkennung (Abs. 5.2.3) ist Hauptverantwortlich für die Initialisierung des Programms. Das Aktualisieren des Modells wird in Abschnitt 5.2.4 sehr ausführlich behandelt.

5.2.1 Template Matching

Das *Template Matching* ist ein Verfahren, welches zum Wiederfinden bekannter Objekte, in einem Bild, verwendet wird. Es hat sich als robustes und einfaches Verfahren in der Objekterkennung, Gesichtserkennung und anderen Bereichen der Bildverarbeitung etabliert. Beim Template Matching werden Referenzbilder (Templates), der zu findenden Objekte erzeugt und in der Regel als Grauwertmatrix gespeichert. Zum Finden des Templates in einem Eingabebild, wird es über alle Bildpositionen geschoben und für jede Position, ein Ähnlichkeitsmaß berechnet. Sobald das Ähnlichkeitsmaß einen definierten Schwellwert erreicht, wird das Bild als gefunden angesehen. Die Bestimmung dieses Schwellwerts wird empirisch durchgeführt. Je nach Anwendung werden alle gefundenen Instanzen oder nur die n besten, Instanzen des Templates zurückgegeben. Template Matching ist translationsinvariant und in speziellen Implementierungen auch rotations- und skallierungsinvariant.

Der im Rahmen dieser Arbeit implementierte Algorithmus basiert auf einer einfachen Form des Template Matching, wobei das Template gegenüber dem Objekt im Eingabebild in Größe und Orientierung nahezu identisch ist. Das Template Matching ist dann nur translationsinvariant. Als Ähnlichkeitsmaß wird die Summe der absoluten Differenzen (SAD) der Grauwerte von Template und Eingabebild verwendet. Sei T das Template mit der Größe $u \times v$ und B das Eingabebild mit der Größe $m \times n$ in dem T gesucht werden soll, dann ist

$$SAD(x,y) = \sum_{i=0}^{u-1} \sum_{j=0}^{v-1} |B(x+i, y+j) - T(i,j)|$$

der SAD an der Stelle (x, y). Algorithmisch lässt sich diese Berechnung als verschachtelte Schleife über die Pixel des Templates darstellen.

```
1 for (int i = 0; i < u; i++) {
2   for (int j = 0; j < v; j++) {
3     bColor = eingabeBild.getColor(x + j, y + i);
4     tColor = template.getColor(j, i);
5     SAD += Math.abs(bColor - tColor);
6   }
7 }
```

Um bestimmen zu können, ob sich das Template an der Stelle $B(x, y)$ befindet, wird, wie bereits erwähnt, ein Schwellwert `minSAD` verwendet. Ist der berechnete SAD kleiner als der definierte minSAD, so wird das Template als gefunden betrachtet.

```
1 for (int i = 0; i < u; i++) {
2   for (int j = 0; j < v; j++) {
3     bColor = eingabeBild.getColor(x + j, y + i);
4     tColor = template.getColor(j, i);
5     SAD += Math.abs(bColor - tColor);
6     if (SAD > minSAD) break;
7   }
8   if (SAD > minSAD) break;
9 }
10 return (SAD < minSAD)?true:false;
```

Die beiden `break`-Befehle beschleunigen den Algorithmus auf eine für PokerBot ausreichende Geschwindigkeit, indem sie die Schleifen gegebenenfalls abbrechen, bevor der SAD für das ganze Template fertig berechnet wurde. Das Eingabebild wird nach dem Template abgesucht, indem durch

$$\sum_{x=0}^{m-u} \sum_{y=0}^{n-v} SAD(x, y)$$

an jeder Stelle $B(x, y)$ die Berechnung des SAD durchgeführt wird und für jede gefundene Instanz die zugehörigen x und y-Werte gespeichert werden(vgl. [5], [20]. Der folgende Quelltext zeigt die Implementierung der `templateMatching`-Methode aus der Bildverarbeitungsklasse der Informationsgewinnung.

```
1 public static ArrayList<Koordinate> templateMatching(BufferedImage
      eingabeBild, BufferedImage template, int minSAD) {
2   if (minSAD == 0) {
3   //wird kein minSAD eingegeben wird anhand des Templates
4   //automatisch eins berechnet.
5     minSAD = template.getHeight() * template.getWidth() * 10;
6   }
7   ArrayList<Koordinate> liste = new ArrayList<Koordinate>();
8   Color color;
9   Color color2;
10  int SAD = 0;
```

```
11   //die Randbereiche werden ignoriert
12   for (int y = 0; y < eingabeBild.getHeight() - template.getHeight(); y
       ++) {
13     for (int x = 0; x < eingabeBild.getWidth() - template.getWidth(); x
       ++) {
14       SAD = 0;
15       for (int i = 0; i < template.getHeight(); i++) {
16         for (int j = 0; j < template.getWidth(); j++) {
17           color = new Color(eingabeBild.getRGB(x + j, y + i));
18           color2 = new Color(template.getRGB(j, i));
19           SAD += Math.abs(color.getRed() - color2.getRed());
20           if (SAD > minSAD) {
21             break;
22           }
23         }
24         if (SAD > minSAD) {
25           break;
26         }
27       }
28       if (SAD < minSAD) {
29         if (liste.isEmpty()) {
30           liste.add(new Koordinate(x, y));
31         } else if ((Math.abs(x - (liste.get(liste.size() - 1).x))) > 2
       || Math.abs(y - (liste.get(liste.size() - 1).y)) > 4) {
32           liste.add(new Koordinate(x, y));
33         }
34       }
35     }
36   }
37   return liste;
38 }
```

Als Eingabeparameter erhält die Methode ein Bild, ein Template, das in dem Bild gesucht werden soll und einen Schwellwert. Zurückgegeben wird eine Liste vom Typ ArrayList<Koordinate>, welche, die Positionen der gefundenen Instanzen enthält. Durch die Abfragen in Zeile 20 wird verhindert, dass die selbe Instanz mehrfach gefunden wird.

5.2.2 Texterkennung

Das stabile Auslesen von Bildschirmtexten ist für die Informationsgewinnung von großer Bedeutung. Es existieren Bibliotheken für Zeichenerkennung (OCR) unter Java. Diese haben häufig das Problem, auf höher auflösende Schriften[5] angewiesen zu sein oder auf die Erkennung von unbekannten Schriften spezialisiert zu sein[6][26]. Die OCR Software *Asprise* funktioniert sehr gut, ist aber keine freie Software[2]. Da zu der Zeit der Implementierung von Pokerbot keine verfügbare Bibliothek den gewünschten Anforderungen entsprach, wurde mit Hilfe des Template Matching eine einfache, aber sehr stabile Zeichenerkennung implementiert. Die Grundidee dieser Zeiche-

[5] Beispielsweise eingescannte Handschrift

nerkennung ist es, eine Liste von Templates als Alphabet zu interpretieren. Jedem Template ist eine Bezeichnung zugeordnet. Beispielsweise wird einem als `A.jpg` gespeichertem Template des Buchstaben *A* der String `A` als Bezeichnung zugeordnet. Eine analoge Zuordnung findet auch bei allen übrigen Buchstaben, Zahlen und Zeichen eines Alphabets statt. Um Text aus einem Bild zu lesen, ist es nur noch notwendig, jedes Element eines Alphabets mit Template Matching über das Bild laufen zu lassen und die Bezeichnungen der gefundenen Instanzen, ihrer Position entsprechend, zu sortieren. Die Methode static String `bildToString(BufferedImage bild, Alphabet alphabet)`(Quelltext 5.6) in der Klasse Texterkennung bietet genau diese Funktionalität. Außerdem stellt die Klasse Texterkennung Methoden zum automatischen Einlesen von Alphabeten bereit.

Quelltext 5.6: Vereinfachte Version der Texterkennung

```
1 public static String bildToString(BufferedImage bild, Alphabet alphabet)
      {
2   //Die Hilfsklasse Zeichen enthällt einen String und eine Koordinate
3   ArrayList<Zeichen> erkannteZeichen = new ArrayList<Zeichen>();
4   ArrayList<Koordinate> xy = new ArrayList<Koordinate>();
5
6   //Schleife über alle Templates eines Alphabets
7   for (int i = 0; i < alphabet.size(); i++) {
8     xy = BV.templateMatching(bild, alphabet.templates.get(i), 0));
9     //Schleife über alle gefundenen Instanzen
10    for (int j = 0; j < xy.size(); j++) {
11      String myString = "" + alphabet.zeichen.get(i);
12      erkannteZeichen.add(new Zeichen(myString, xy.get(j)));
13    }
14    xy.clear();
15  }
16  return sortiereNachPosition(erkannteZeichen); //pseudocode
17 }
```

Erstellung von Templates

Insbesondere bei der Schrifterkennung hat die Auswahl der richtigen Templates eine große Auswirkung auf die Qualität des Template Matching. Um Fehler zu vermeiden, müssen einige Aspekte bei der Erstellung von Templates beachtet werden:

- Das Template muss so gewählt werden, dass ein zu erkennendes Objekt eindeutig identifiziert werden kann.
- Ein zu kleines Template kann dazu führen, dass ein Objekt, fälschlicher Weise, in anderen Bildbereichen erkannt wird.
- Bildelemente, die nicht zur Identifizierung beitragen, sollten nicht in das Template einbezogen werden. Eine möglichst geringe Größe vermeidet unnötigen Rechenaufwand und steigert die Qualität.

Abbildung 5.1: Template der Ziffer Drei

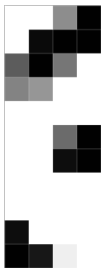


Abbildung 5.2: Karte Pik Drei

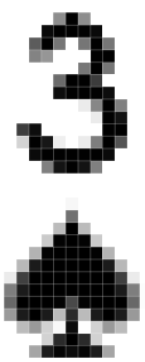


Abbildung 5.1 zeigt ein Template für die Erkennung der Ziffer Drei. Das Template wird durch die geringe Größe schnell gefunden und repräsentiert eindeutig das gesuchte Zeichen. Mit diesem Template kann die Wertigkeit der Karte aus Abbildung 5.2 ermittelt werden.

Für eine Erkennung einer transparenten Schrift, auf einem nicht einfarbigen Hintergrund, sind zusätzlich folgende Punkte zu beachten:

- Das Template muss so gewählt werden, dass möglichst wenige Pixel des Randbereichs enthalten sind. Die Farbunterschiede am Rand sind oft zu groß, für eine fehlerlose Erkennung.
- Wenn ein Template zu viele Pixel der Hintergrundfarbe beinhaltet, können mehrere Templates, mit beispielsweise unterschiedlichen Hintergründen, für dasselbe Zeichen erstellt werden. Dabei müssen die Templates in etwa denselben Bereich eines Zeichens beinhalten. Andernfalls ist es möglich, dass ein Zeichen mehrfach erkannt wird.

In Abbildung 5.3 sind die Bereiche die ein Template bilden Könnten markiert. Die Ziffern wurden so ausgeschnitten, dass möglichst viele Pixel enthalten sind, die nicht von dem Hintergrund beeinflusst werden. Das Komma beinhaltet einen großen Ausschnitt des Hintergrundes, damit das Template nicht zu klein wird. Ohne die Einbeziehung des Hintergrundes, würde

Abbildung 5.3: Auswahl von Templates

es zu einer fehlerhaften Erkennung kommen, da die Pixel eines Kommas auch in Ziffern enthalten sind. Hier bietet es sich an, mindestens zwei Templates eines Kommas mit unterschiedlichen Hintergrundfarben zu erstellen.

5.2.3 Fenstererkennung

Der in Abschnitt 5.1.3 besprochene Initialisierungsprozess verwendet die bisher nicht vorgestellte Methode `Fenstererkennung.getSichtbareFenster( bwScreenShot, bot.getPokerPlattformen(),i)` (siehe Programm 5.5 Z. 9). Diese Methode findet offene Pokerfenster aller übergebenen Pokerplattformen, indem das über `pokerPlattform.getLogo()` geladene Template, mittels Template Matching, im ebenfalls übergebenen Screenshot, gesucht wird. Anschließend werden Größen und Positionen aller gefundenen Fenster ermittelt und die Fenster schließlich zurückgegeben. Der nachstehende Quelltext zeigt die Methode[6] vollständig.

```
1 public static ArrayList<Fenster> getSichtbareFenster(BufferedImage
      screenshot, ArrayList<Pokerplattform> pokerPlattformen, int device)
      {
2   ArrayList<Fenster> fenster = new ArrayList<Fenster>();
3   for (int x = 0; x < pokerPlattformen.size(); x++) {
4     fenster.addAll(getSichtbareFenster(screenshot, pokerPlattformen.get(
      x), device));
5   }
6   return fenster;
7 }
8
9 public static ArrayList<Fenster> getSichtbareFenster(BufferedImage
      screenshot, Pokerplattform pokerPlattform, int device) {
10  ArrayList<Fenster> fensterListe = new ArrayList<Fenster>();
11  BufferedImage logoTemplate = pokerPlattform.getLogo();
12  ArrayList<Koordinate> gefundeneLogos;
13  gefundeneLogos = BV.templateMatching(screenshot, logoTemplate, 100);
```

[6] Die Methode ist, wegen der besseren Überschaubarkeit, in zwei überladene Methoden aufgeteilt.

```
14
15   for (int i = 0; i < gefundeneLogos.size(); i++) {
16     Fenster tempFenster = new Fenster(gefundeneLogos.get(i).add(
       pokerPlattform.getLogoOffset()), pokerPlattform, device);
17     fensterListe.add(tempFenster);
18   }
19   return fensterListe;
20 }
```

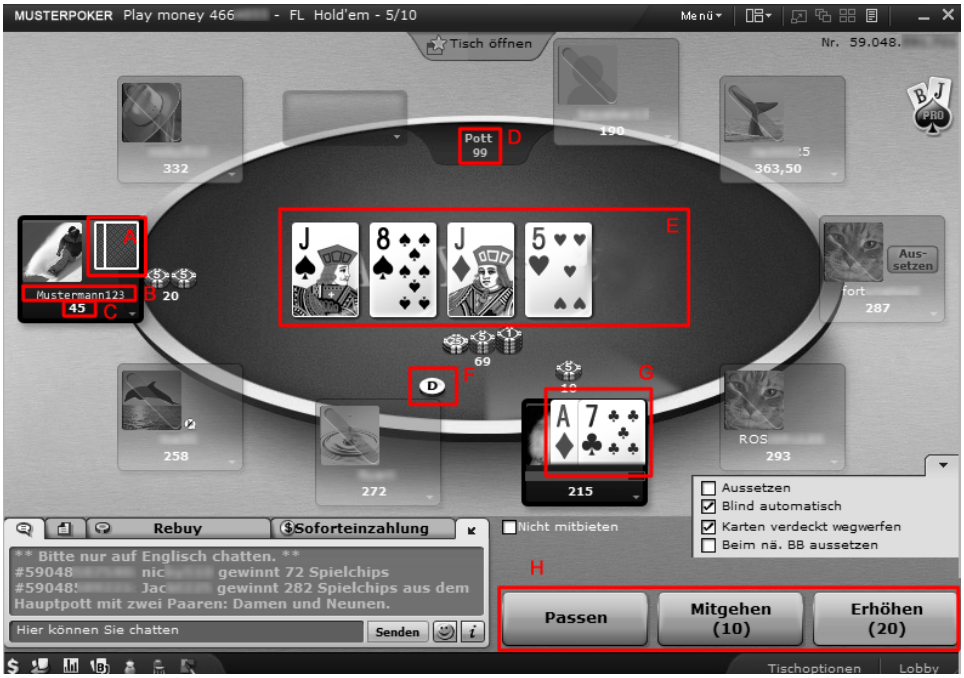
5.2.4 Aktualisierung des Spielzustandsmodells

Immer, wenn PokerBot an der Reihe ist, einen neuen Zug zu machen, wird vorher das in Abschnitt 4.2.2 vorgestellte Modell aktualisiert. Dazu wird die Methode `aktualisiereModell(Spiel spiel, BufferedImage screen)` in der Hauptschleife aufgerufen(vgl. Abs. 5.1.4 Quelltext Z.12-14). Die Aufgabe dieser Methode ist es, das im Parameter `Spiel spiel` übergebene Modell(vgl. Abs. 5.1.2) um die aktuelle Runde zu erweitern. Eine neue Runde wird also erstellt und dessen Variablen durch einzelne Methodenaufrufe gefüllt. Zunächst geschieht gegebenenfalls das Inkrementieren der Variablen: `spiel`, `hand`, `phase`, und (Wett)`runde`.

```
1 public static void aktualisiereModell(Spiel spiel, BufferedImage screen)
      {
2
3   Runde aktuelleRunde = new Runde();
4   Runde letzteRunde = null;
5
6   if (spiel.modell.isEmpty()) {
7     aktuelleRunde.spiel = getSpiel();
8     aktuelleRunde.hand = 1;
9     aktuelleRunde.phase = Phase.PREFLOP;
10    aktuelleRunde.runde = 1;
11  } else {
12    letzteRunde = spiel.modell.get(spiel.modell.size() - 1);
13    aktuelleRunde.spiel = letzteRunde.spiel;
14    aktuelleRunde.phase = getPhase(spiel, screen);
15    aktuelleRunde.hand = getHand(spiel, aktuelleRunde, letzteRunde,
      screen);
16    aktuelleRunde.runde = getRunde(spiel, aktuelleRunde, letzteRunde);
17  }
18  ...
19 }
```

Anschließend findet die Extraktion von Informationen aus dem übergebenen Screenshot statt. Abbildung 5.4 veranschaulicht, welche Informationen zu extrahieren sind. Um den Umgang mit den unterschiedlichen Situationen zu vereinfachen, ruft die besprochene Methode entweder `aktualisiereModellNeueHand(...)` oder `aktualisiereModellJedeHand(...)` auf. Die Methodennamen entsprechen den jeweiligen Situationen, in denen sie aufgerufen werden. Die in Quelltext 5.7 zu sehenden Programmzeilen

Abbildung 5.4: Screenshot des aktuellen Spielzustands. Die Bereiche aus denen Informationen extrahiert werden sind rot markiert. Bereiche die bei jedem Spieler überprüft werden (A, B, C und F) sind nur einmal exemplarisch markiert.



sind aus den beiden zuletzt genannten Methoden entnommen und sollen einen Eindruck von der Funktionsweise derselben vermitteln. Kommentare am Ende einer Zeile stellen die Verbindung zu Abbildung 5.4 her. Zu erwähnen ist, dass der Parameter `spiel` an fast alle Methoden mit übergeben wird. Das liegt daran, dass je nach Pokerplattform unterschiedliche Implementierungen der einzelnen Methoden möglich sind. Im richtigen Quelltext wird häufig direkt `spiel.fenster.pokerPlattform` übergeben. Durch diesen Parameter wird beeinflusst, welche Implementierung gewählt wird.

Quelltext 5.7: Auszüge der Aktualisierungsmethoden

```
1 //Beispiele aus aktualisiereModellNeueHand(...)
2 aktuelleRunde.handKarten = Kartenerkennung.getHandKarten(screen, spiel)
      ;//G
3 aktuelleRunde.pot = Statuserkennung.getPot(screen, spiel);//D
4 aktuelleRunde.dealerPos = Statuserkennung.dealerPosition(screen, spiel)
      ;//F
5
6 //Schleife über alle Plätze am Tisch
7 for (int i = 0; i < spiel.pokerPlattform.getMaxPlayers(); i++) {
8   //if Platz besetzt
9   Spieler neuerSpieler = new Spieler();
```

```
10   double stack = GegnerAnalyse.getSpielerStack(i, screen, spiel);//C
11   neuerSpieler.setStack(stack);
12   ...
13    //ein leeres Array 'spieler' wurde erzeugt
14     aktuelleRunde.spieler[i] = neuerSpieler;
15 }
16
17 //Beispiel aus aktualisiereModellJedeRunde(...)
18 if (aktuelleRunde.runde == 1) {
19   aktuelleRunde.gemKarten = Kartenerkennung.getGemKarten(screen, spiel)
       ;//E
20 } else {
21   aktuelleRunde.gemKarten = letzteRunde.gemKarten;
22 }
```

Informationsgewinnung über Template Matching

Alle in Abbildung 5.4 durch die Buchstaben A-H markierten Elemente werden mit Template Matching analysiert. Für jedes Element sind folgende drei Schritte durchzuführen:

1. Herausfinden, wie das gesuchte Objekt aussieht und wo es sein kann.
2. Überprüfen, an welcher Position sich das Objekt befindet.
3. Erkenntnisse in das Modell schreiben.

Je nach Methode sind die einzelnen Schritte unterschiedlich. Der erste Schritt erfordert meistens das Auslesen von Templates, Alphabeten und Koordinaten aus den der vom Interface Pokerplattform implementierenden Klassen. Der zweite Schritt benutzt diese Informationen um durch Template Matching herauszufinden, wo die Objekte sind. Im dritten Schritt wird überprüft, was ein Objekt an der gefundenen Position bedeutet und Entsprechendes dann in das Modell eingetragen. Quelltext 5.8 zeigt die drei Schritte an dem Beispiel der Bestimmung, der Position, des Kartengebers(F). Bei den übrigen Elementen wird auf ähnliche Weise verfahren.

Quelltext 5.8: Beispiel für Informationsgewinnung

```
1 public static int dealerPos(BufferedImage screen, Pokerplattform pp) {
2   BufferedIImage db = pp.getDealerbuttonTemplate();  //Schritt 1
3   for (int platz = 0; platz < pp.getMaxPlayers(); platz++) { //Schritt 1
4     Koordinate xy = pp.getDealerbuttonPos(platz); //Schritt 1
5     screen = screen.getSubimage(xy.x, xy.y, db.b, db.h); //Schritt 1
6     if (BV.templateMatching(screen, db, 0)) return platz; //Schritt 2
7   }
8   return 0;
9   // Schritt 3 geschieht durch den Aufruf dieser Methode.
10 }
```

Füllen der restlichen Variablen des Modells

Nicht alle Informationen sind direkt auf einem Screenshot des Spielzustands zu erkennen. Dazu zählen beispielsweise die aktuelle Phase, die Nummer der Hand oder die Wettrunde. Auch welcher Spieler welche Aktion gemacht hat, ist nicht explizit dargestellt. Während ein menschlicher Spieler diese Informationen leicht aus dem vorliegenden Bild schlussfolgern kann, ist diese Aufgabe für Pokerbot nicht trivial. Durch die Anzahl der Karten, die in der Tischmitte liegen ist die Phase zu erkennen. Die Wettrunde und Hand kann nach jeder Aktion von PokerBot inkrementiert und beim Eintritt in eine neue Hand, bzw. Phase, auf Null gesetzt werden.

Aktionen der Mitspieler

Ein Rückschluss auf die Aktionen der Mitspieler ist nicht so leicht. Es muss beachtet werden, wie viel ein Mitspieler schon vor der Runde gesetzt hat und wie viel er hätte setzen müssen, um zu erhöhen oder mitzugehen. Dazu werden alle Kontostände, jedes Mal, wenn PokerBot an der Reihe ist, analysiert. Hier können Sonderfälle auftreten. Es entsteht ein Informationsdefizit bei dieser Analyse, wenn die erste Entscheidung eines Spiels getroffen werden muss oder ein neuer Spieler dem Tisch beitritt.

Folgendes Szenario soll die Problematik verdeutlichen: Angenommen zu Beginn einer neuen Hand tritt ein neuer Mitspieler einem Tisch bei. Der erste Screenshot wird aufgenommen, sobald PokerBot in dieser neuen Hand an der Reihe ist. Falls der neue Spieler schon vor PokerBonz zum aktuellen Kontostand gebildet werden kann. Dieser Sonderfall wird durch eine spezielle Behandlung der ersten Wettrunde einer Hand abgefangen. Dabei werden dit an der Reihe war und eine Aktion gemacht hat, lässt sich diese nicht über den Kontostand ermitteln, da der alte Kontostand unbekannt ist und so keine Differee gesetzten Beträge aus der Mitte des Tisches, beginnend bei dem Small-Blind, ausgelesen. Durch die gesetzten Beträge werden die Aktionen ermittelt. Es ist zu beachten, dass das Auslesen, der gesetzten Beträge, aus der Tischmitte, in späteren Wettrunden nicht immer möglich ist, da die Informationen von anderen Tischobjekten, wie etwa gegnerische Jetons oder Gemeinschaftskarten, verdeckt werden können.

Eine weitere Schwierigkeit besteht darin, dass nach dem Zug von PokerBot die aktuelle Wettrunde nicht zwingend zu Ende ist. Demnach müssen nicht nur die Aktionen der Spieler aus der aktuellen Runde, sondern auch jene aus der letzten Runde erschlossen werden, um das Modell vollständig zu halten.

Die vollständige Ermittlung der gegnerischen Aktionen erstreckt sich über viele Seiten Quelltext und ist deshalb direkt dem im Rahmen dieser Arbeit entwickelten Programm[7] zu entnehmen.

[7] Der entsprechende Code befindet sich in Paket: bildverarbeitung, Klasse:

5.3 Modul 2: Spiellogik

Das Spiellogikmodul ist dafür zuständig, aus dem Modell des aktuellen Spielzustands zu ermitteln, welche Aktion gemacht werden soll. Wie in Abbildung 4.4 zu sehen, lässt sich das Modul in mehrere Schritte unterteilen. Die Methode `Aktion aktionsBestimmung(...)`, die aus der Hauptschleife aufgerufen wird, verwaltet intern die Aktionsbestimmung. Drei Methoden werden dazu verwendet:

1. `Aktion getAktion(...)` - Ermittelt eine Aktion nach der Strategie von www.Pokerstars.com. Die Funktionalität dieser Methode ist in zwei Teilschritte zu trennen:
   - Interpretation der Daten aus dem Modell des Spielzustands.
   - Bestimmung der Aktion, die in der ermittelten Situation zu machen ist.
2. `Aktion pruefeAktion(...)` - Prüft, ob die vorgeschlagene Aktion mathematischen Prinzipien entspricht. Die Art der Überprüfung kann über die grafische Benutzeroberfläche angepasst werden.
3. `Aktion getGueltigeAktion(...)` - Überprüft, ob die vorgeschlagene und mathematisch überprüfte Aktion korrekt und sinnvoll ist.

5.3.1 Interpretation des Modells

Als Vorarbeit für die Anwendung, des in Abschnitt 4.2.3 angesprochenen Regelsystems, werden zunächst die im Model enthaltenen Daten interpretiert. Das Regelsystem erwartet eine Anfrage, die folgende Informationen enthält (vgl. Abs. 5.1.2):

- Phase
- Karten
- Blatt/PreFlopBlatt
- Position
- GegnerAktion
- Pot

`Phase`, `Karten` und `Pot` können direkt aus dem Model ausgelesen werden. Auch die Bestimmung der `Position` stellt keine große Herausforderung dar. Es genügt den Abstand der eigenen Position von der Position des Kartengebers zu errechnen. Dabei sollten nur Spieler beachtet werden, die zu Beginn der Runde noch Karten hatten.

Die `GegnerAktion` lässt sich ähnlich bestimmen. Eine Schleife über alle Spieler zählt die Aktionen. Anschließend wird entsprechend ein Element des Enums `GegnerAktion` zurückgegeben.

`GegnerAnalyse`, Methode: `setSpielerAktionen(...)`

Bestimmen von PreFlopBlatt

Im Preflop unterteilt das verwendete Regelsystem die Blätter in Kategorien (siehe Abs. 3.2). Diese entsprechen den Elementen des Enums `PreFlopBlatt`.

```
1 public enum PreFlopBlatt {
2   SEHRSTARK, STARK, MITTEL, SPEKULATIV, GEMISCHT, SCHLECHT
3 };
```

Die Einteilung der Blätter erfolgt manuell nach dem Schema, wie es aus dem Quelltext 5.9 ersichtlich ist. Für jede mögliche Starthand wird mittels einfacher Abfragen getestet, ob es sich bei den Karten im Array `karte` um die entsprechende Starthand handelt.

Quelltext 5.9: Ermitteln des PreflopBlatts.

```
1 //T9s
2 if ((karte[0].getWert() == Wert.ZEHN && karte[1].getWert() == Wert.NEUN
3     || karte[1].getWert() == Wert.ZEHN && karte[0].getWert() == Wert.
      NEUN)
4     && karte[0].getFarbe() == karte[1].getFarbe()) {
5   return PreFlopBlatt.GEMISCHT;
6 }
7
8 //88 77 66 55
9 if ((karte[0].getWert() == Wert.SIEBEN
10     || karte[0].getWert() == Wert.SECHS
11     || karte[0].getWert() == Wert.FUENF
12     || karte[0].getWert() == Wert.ACHT)
13     && karte[0].getWert() == karte[1].getWert()) {
14   return PreFlopBlatt.SPEKULATIV;
15 }
```

Das Blatt

Ähnlich wie bei den Preflopblättern unterteilt das Regelsystem auch alle anderen Blätter in Kategorien. Auch hier werden die einzelnen Kategorien durch Enums realisiert:

```
1 public enum Blatt {
2   NICHTS, COMMUNITYWIN, UEBERGUTSHOT, UEBERKARTEN, DOPPELGUTSHOT,
      GUTSHOT, OESD, FLUSHDRAW, MONSTERDRAW, PAAR, ZWEIPAAR, DRILLING,
      STRASSE, FLUSH, FULLHOUSESCHWACH, FULLHOUSE, VIERLING, STRAIGHTFLUSH
3 };
```

Selbst diese große Anzahl unterschiedlicher Blätter reicht nicht aus, um alle möglichen Fälle zu klassifizieren. Die Klasse `BlattWert` ermöglicht eine weitere Unterteilung der einzelnen Blätter. Jedes Objekt von `BlattWert` besteht aus einem Blatt und einer variablen Anzahl an Parametern. Bei einem Flush wird durch die Parameter beispielsweise angegeben, welchen Wert die höchste Karte des Flush's hat und ob diese Karte eine Gemeinschaftskarte oder eine Handkarte ist.

Den Blattwert bestimmen

Die Methode `BlattWert getMeinBlattWert(Karte[] handKarten, Karte[] gemKarten)` ist dafür zuständig aus den übergebenen Karten das beste Blatt zu ermitteln und dieses als Objekt der Klasse `BlattWert` zurückzugeben. Dazu werden nacheinander Methoden aufgerufen, die jeweils ein Blatt erkennen können. So gibt es zum Beispiel eine Methode für die Erkennung eines Flush's, eine für die Erkennnung einer Straße und so weiter. Diese Methoden erwarten als Eingabeparameter allerdings nicht die Karten, sondern Metainformationen über die Karten.

Die Metainformationen werden generiert, indem die Karten an den Konstruktor der Klasse `KartenArrays` übergeben werden. Anschließend stehen zwei Arrays vom Typ `int` für die Handkarten und zwei für die Gemeinschaftskarten zur Verfügung. Jeweils ein Array mit Daten über die Werte der Karten und eins mit Daten über die Farben. Die Arrays sind zunächst gefüllt mit Nullen und haben 13[8] beziehungsweise vier[9] Stellen. Für jede übergebene Karte wird in einem Werte-Array und in einem Farben-Array an einer Stelle inkrementiert. Folgendes Beispiel hilft bei der Verständnis der Blatterkennung. Welches Blatt lässt sich in den Metadaten erkennen?

```
1 //A,K,Q,J,T,9,8,7,6,5,2,1
2 int[] handWerte = {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1};
3 int[] handFarben = {2, 0, 0, 0};
4 int[] gemWerte = {0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1};
5 int[] gemFarben = {3, 0, 1, 0};
```

Zwei Handkarten und drei Gemeinschaftskarten haben die gleiche Farbe. Dadurch ist zu erkennen, dass es sich bei dem Blatt um einen Flush handelt. Ohne die Farben zu betrachten, könnte man ein Paar Zweien, mit einem Ass als Kicker, erkennen. Nach diesem Prinzip funktionieren auch die erwähnten Methoden.

Tests auf bestimmte Blätter

Sobald die Metadaten zur Verfügung stehen, werden die Methoden zur Überprüfung bestimmter Blätter aufgerufen. Die Reihenfolge der Aufrufe entspricht der Stärke der Blätter, angefangen bei dem Stärksten. Dadurch werden unnötige Methodenaufrufe vermieden. Exemplarisch wird in Quelltext 5.10 die Methode zur Erkennung eines Full House vorgestellt. Diese Methode ist eine der simpelsten ihrer Art. Die Überprüfung auf einen Flush umfasst über 100 Zeilen Quelltext. Der komplette Quelltext ist unter [14] zu finden.

[8] für die Werte
[9] für die Farben

Quelltext 5.10: Diese Methode überprüft, ob ein Full House gebildet werden kann und gibt ein entsprechendes Objekt der Klasse `BlattWert` zurück. Die Parameter `wert1` und `j` geben Auskunft darüber, welches Full House es gegebenenfalls ist.

```
1  static BlattWert testAufFullHouse(int[] _hand, int[] _gem) {
2    int[] hand = _hand.clone();
3    int[] gem = _gem.clone();
4    int wert1;
5    for (int i = 12; i >= 0; i--) {
6      if ((hand[i] + gem[i]) >= 3) {
7        wert1 = i;
8        hand[i] = 0;
9        gem[i] = 0;
10       for (int j = 12; j >= 0; j--) {
11         if ((hand[j] + gem[j]) >= 2) {
12           return new BlattWert(Blatt.FULLHOUSE, wert1, j);
13         }
14       }
15     }
16   }
17   return new BlattWert(Blatt.NICHTS, -1);
18 }
```

5.3.2 Formalisierung des Regelsystems

In Abschnitt 3.2 wurde eine Strategie vorgestellt, bei der ein menschlicher Spieler Regeln befolgt, die ihm sagen welche Aktion er in welcher Situation, durchzuführen hat. Zur Implementierung dieses Systems müssen die Regeln formalisiert sein.

Regeln im Preflop

In Abschnit 3.2 wurde das Spielen vor dem Flop behandelt. Für jede Situation, die in der Preflopphase auftreten kann, gibt es entsprechende Tabellen, aus denen die richtigen Aktionen entnommen werden können. Ein Beispiel für die Formalisierung der vorgestellten Tabelle 3.3 ist in Quelltext 5.11 zu sehen.

Quelltext 5.11: Formalisierung der Starthandtabellen aus dem Regelsystem von Pokerstrategy.com.

```
1  switch (preflopBlatt){
2  ...
3  case MITTEL:
4    switch (position) {
5      case FRUEH:
6        if (gegnerAktion == EINERHOCHUNDEINERMIT &&
7          //Karten == KQs)
8          return Aktion.CALL;
9        else
10         return Aktion.CHECK;
```

```
11      case MITTE:
12        if (gegnerAktion == ALLERAUS ||
13          gegnerAktion == EINERMIT)
14          return Aktion.RAISE;
15        if (gegnerAktion == EINERHOCHUNDEINERMIT &&
16          //Karten == KQs)
17          return Aktion.CALL;
18        else
19          return Aktion.CHECK;
20      case SPAET:
21        ...
```

Regeln nach dem Flop

Das Vorgehen für die Formalisierung des Regelsystems nach dem Flop ist dasselbe, wie vor dem Flop. Durch viele verschachtelte Abfragen werden die vor verarbeiteten Daten soweit durchsucht, bis die aktuelle Situation genau bestimmt ist und eine Aktion feststeht. Dabei wird zuerst die Phase und anschließend der Blattwert beachtet. Je nach Blattwert werden dann weitere Informationen, wie die Anzahl der aktiven Spieler oder Aktionen von Mitspielern hinzugezogen. Das folgende Beispiel (aus Abschnitt 3.2) ist eine Regel für das Verhalten nach dem Flop:

> Falls du einen Gutshot oder Überkarten hast gilt:
>
> - Du setzt nur, wenn du vor dem Flop erhöht hast und maximal zwei Gegner hast.
> - Du gehst einen gegnerischen Einsatz oder eine Erhöhung nur mit, wenn der Pot mindestens das zehnfache des zu bringenden Einsatzes enthält

Der dazugehörige Quelltext 5.12 zeigt auf, wie die Regeln auf Programmcode-Ebene formuliert sind.

Quelltext 5.12: Regeln nach dem Flop.

```
1  switch (phase){
2  ...
3  case FLOP:
4    switch(blatt){
5    case GUTSHOT:
6    case UEBERKARTEN:
7      if(gegnerAktion == ALLECHECK)
8        if(letzteRunde.eigeneAktion == RAISE &&
9          aktuelleRunde.anzahlGegner < 3)
10         return Aktion.RAISE;
11       else
12         return Aktion.FOLD;
13     else
14       if(aktuelleRunde.pot >= aktuelleRunde.Spieler[Bot].toCall * 10)
15         return Aktion.CALL;
```

Problematik bei der Implementierung des Regelsystems

Es stellt sich heraus, dass der Code zwar gut lesbar und auch leicht zu programmieren ist, jedoch gibt es sehr viele Sonderfälle, die zu behandeln sind. An zahlreichen Stellen bietet die Anleitung von Pokerstrategy.com viel Interpretationsspielraum. Für einen Menschen ist sehr leicht zu entscheiden, welche Aktion sinnvoll ist, falls das Regelsystem einen speziellen Fall nicht abdeckt. Für die Implementierung der Anleitung müssen aber alle Fälle die Auftreten können behandelt werden. Die Menge und die Komplexität dieser Sonderbehandlungen übersteigen bei Weitem den Rahmen in dem alle Regeln überblickt werden können, wodurch das Treffen falscher Entscheidungen nicht ausgeschlossen werden kann.

Folgendes Szenario soll einen Eindruck davon vermitteln, welche Art von Problemen auftreten kann. Bei einem Drilling, der aus der kleinsten Gemeinschaftskarte und einem Paar auf der Hand gebildet wird, entscheidet die implementierte KI nur zu erhöhen, wenn kein Flush gebildet werden kann. Falls doch ein Flush möglich ist und ein gegnerischer Spieler eine beliebige Erhöhung macht, so wird von der KI nicht weiter erhöht. Dabei ergeben weniger als fünf Prozent aller möglichen Starthände ein stärkeres Blatt, als den Drilling. Außerdem wird der Drilling mit jeder weiteren aufgedeckten Karte schwächer, da mehr Karten dem Flushdraw helfen, als dem fertigen Drilling.

Eine Vielzahl von Faktoren erschwert die Entscheidungsfindung im aufgezeigten Szenario. So muss, etwa bei einem Drilling, überprüft werden, ob durch die Gemeinschaftskarten ein besseres Blatt gebildet werden kann. Zu prüfen wäre hier, ob ein Flushdraw oder ein Straightdraw vorliegt. Weiterhin müssten die Draws daraufhin getestet werden, ob die eigene Hand von diesem Draw profitieren könnte. Ein wichtiges Detail ist auch, ob sich der Drilling durch ein Paar in den Gemeinschaftskarten oder in den Handkarten gebildet hat. Ein Drilling durch ein Paar in den Handkarten hat einen größeren Wert, da ein gegnerisches Full House ausgeschlossen werden kann. Bei jeder Eventualität müssen die Aktionen der Mitspieler im Auge behalten werden, was die Verzweigung der möglichen Fälle weiter vertieft und den Programmcode sehr stark wachsen lässt.

5.3.3 Stochastische Überprüfung

Ein Lösungsweg die Fehler aus der starren KI abzufangen, ist eine allgemeine mathematische Überprüfung. Das ist die Aufgabe der Methode: `Aktion pruefeAktion(...)`.

Gewinnwahrscheinlichkeit errechnen

Im Poker gibt es 1326 verschiedene Kombinationen von Handkarten. Jede dieser Kombination hat die Möglichkeit auf einen Gewinn. Durch die Verbindung von Handkarten und Gemeinschaftskarten entstehen verschieden starke

Blätter. Die Grundidee der stochastischen Überprüfung ist die Gegenüberstellung der eigenen Handkarten zu allen möglichen Handkarten.

Die Gegenüberstellung läuft folgendermaßen ab:

- Aus allen noch unbekannten Karten werden alle möglichen Kartenpaare gebildet. Sei n die Menge aller noch unbekannten Karten, dann ist $i = n^2 - n/2$ die Menge der Kartenpaare.
- Anschließend wird gezählt, wie viele dieser Kombinationen, in Verbindung mit den Gemeinschaftskarten, ein schlechteres Blatt als das Eigene ergeben. Sei A das eigene Blatt und B ein Kartenpaar der Menge i, dann sind i Vergleiche der Form $schlechter(A, B)$ durchzuführen.
- Die Summe der schlechteren Hände, geteilt durch die Anzahl aller gebildeter Kombinationen i, ergibt die Wahrscheinlichkeit, zum Zeitpunkt der Überprüfung, nicht das beste Blatt zu haben. Falls mehr als ein anderer Spieler im Spiel ist, wird diese Wahrscheinlichkeit mit der Anzahl der Spieler s multipliziert und ergibt die Wahrscheinlichkeit Q. Diese ist abhängig von der Anzahl der Mitspieler:
- Die eigenen Karten gewinnen demnach mit der Wahrscheinlichkeit:

$$P = 1 - Q = 1 - (\frac{s}{i} \sum_{x=0}^{i} schlechter(A, B))$$

Echte Gewinnwahrscheinlichkeit

Mit einem Draw oder Überkarten im Flop, ist die Wahrscheinlichkeit aktuell die beste Hand zu haben sehr klein. Die Chancen, dass sich diese Hand verbessert sind aber größer, als die Chancen vieler im Flop noch stärkerer Kartenpaare. Um eine korrekte Gewinnwahrscheinlichkeit zu ermitteln, müssen also die noch kommenden Karten mitberücksichtigt werden. Dazu müsste P, bei n unbekannten Karten im Flop $n * n - 1$ mal berechnet werden. Insgesamt wären das also $i * n * n - 1$ Vergleiche. Bei 47 unbekannten Karten sind das 2337122 Methodenaufrufe.

Vergleich von Blättern

In diesem Unterabschnitt wird die Methode `int aStaerkerB(...)` vorgestellt. Sie wird verwendet, um zu prüfen, welches von zwei übergebenen Kartenpaaren A und B, mit den ebenfalls übergebenen Gemeinschaftskarten, das bessere Blatt bildet. Zurückgegeben wird 1, falls A stärker ist, 2, falls B stärker ist und 0 falls beide gleich stark sind. In Quelltext 5.13 ist zu sehen, wie die Methode funktioniert. Zum Vergleichen wird die Klasse `BlattWert` verwendet.

Quelltext 5.13: Die Methode vergleicht zwei Kartenpaare.

```
1 public static int aStaerkerB(Karte[] _A, Karte[] _B, Karte[] gemKarten)
      {
2   //Rückgabe ist 1 wenn A stärker, 2 wenn B stärker, ansonsten 0.
3   BlattWert A = PokerLogik.blattWert(_A, gemKarten);
4   BlattWert B = PokerLogik.blattWert(_B, gemKarten);
5
6   if (A.blatt.ordinal() > B.blatt.ordinal()) {
7     return 1;
8   } else if (A.blatt.ordinal() < B.blatt.ordinal()) {
9     return 2;
10  } else {
11    //Beide Blätter sind vom selben Typ:
12    if (A.blatt == Blatt.VIERLING) {
13    //Beide Blätter sind vom Typ Vierling
14    //wert ist die Höhe des Vierlings
15      if (A.wert > B.wert) {
16        return 1;
17      } else if (A.wert < B.wert) {
18        return 2;
19      } else {
20        //wert2 ist der Kicker
21        if (A.wert2 > B.wert2) {
22          return 1;
23        } else if (A.wert2 < B.wert2) {
24          return 2;
25        } else {
26          return 0;
27    ...
28    //bei anderen Blättern ist das Vorgehen analog zum Vierling
```

Reduktion der Methodenaufrufe durch Monte Carlo Simulation

Monte Carlo Simulation ist ein Sammelbegriff für Verfahren, die schwer zu berechnende Probleme mit dem Gesetz der großen Zahlen lösen. Es wird davon ausgegangen, dass Probleme die viele Rechenschritte benötigen, durch die Berechnung einer geeigneten Stichprobe dieser Rechenschritte, hinreichend angenähert werden können [28]. PokerBot bedient sich dieser Idee, indem von den bis zu 2337122 Methodenaufrufen nur ein kleiner, zufällig gewählter, Anteil tatsächlich aufgerufen wird.

Pot Odds

Die Pot Odds werden nur errechnet, wenn die stochastische Überprüfung zu dem Ergebnis kam, dass die Aktion Schieben beziehungsweise Aussteigen getätigt werden soll und die aktuelle Phase entweder Flop oder Turn ist. Die ursprüngliche Implementierung hatte hier nur das Testen auf Draws vorgesehen. Diese Vorgehensweise ist jedoch zu fehleranfällig, da die Ergebnisse zu grob sind. Die erste Implementierung der Out Zählung wurde nach menschlichen Vorgehensweisen entwickelt. Dabei wird jede mögliche Karte

daraufhin überprüft, ob durch sie das eigene Blatt soweit verbessern konnte, dass es stärker war, als andere mögliche Blätter. Wenn zum Beispiel bereits ein Flushdraw einer Farbe in den Gemeinschaftskarten lag, wurden nur Karten als Out gezählt, welche auch mindestens zu einem Flush führten. Weiterhin wurde eine Karte auch nicht als Out gezählt, wenn das Blatt der Gemeinschaftskarten verbessert wurde. Das Problem war der Konflikt, ob eine Karte dem Blatt der Gemeinschaftskarten mehr helfen würde, als dem Eigenen. Wenn Beispielsweise eine bestimmte Karte ein Paar in den Gemeinschaftskarten entstehen ließe, jedoch einen Flush mit den Handkarten erzeugen würde, würde sie aufgrund der Verbesserung der Gemeinschaftskarten nicht gezählt werden. Diese Zählweise bestimmt daher die Wahrscheinlichkeit auf einen Gewinn, in einigen Fällen, sehr ungenau. Die genaueste Lösung liefert das Zählen der Outs mithilfe der errechenbaren Gewinnwahrscheinlichkeit, welche im Absatz 5.3.3 erläutert wird. So wird jede mögliche Karte als Out gezählt, welche, in Kombination mit den aktuellen Handkarten und Gemeinschaftskarten, eine sehr hohe Gewinnwahrscheinlichkeit erreicht. Die Berechnung der Pot Odds ist trivial, falls die Anzahl der Outs bekannt ist und wurde, genau nach der Erklärung in Absatz 3.1.3, implementiert.

5.3.4 Nachkontrolle

Dadurch, dass sowohl die menschliche Pokerstrategie, als auch die mathematische Kontrolle heuristische Verfahren anwenden, werden bestimmte Situationen nicht korrekt behandelt. Der erste Teil des Spiellogikmoduls entscheidet, bei sehr schlechten Karten immer rauszugehen. Auch die mathematische Überprüfung ändert nichts an dieser Entscheidung. In beiden Fällen wird die Untersuchung vieler weiterer Faktoren, allein aufgrund der schlechten Karten, nicht durchgeführt.

Das bedeutet, es wird gar nicht geprüft, was die anderen Spieler gemacht haben. Oft kommt es vor, dass kein Spieler erhöht hat. In einem solchen Fall kann PokerBot im Spiel bleiben, ohne einen Einsatz zu machen. Nach den vorherigen Schritten der Pokerlogik würde PokerBot aber dennoch aussteigen. Neben diesem Fall kann es auch vorkommen, dass PokerBot erhöhen soll, obwohl die maximale Anzahl an Erhöhungen schon stattgefunden hat. Diese und ähnliche Fälle werden von der Nachkontrolle behandelt.

Die Implementierung der Nachkontrolle ist unkompliziert. Durch eine Reihe von Vergleichen der vorgeschlagenen Aktionen und der möglichen Aktionen wird immer die am besten passende Aktion ausgewählt. Quelltext 5.14 zeigt einen Ausschnitt der Methode `Aktion getGueltigeAktion(...)`, die als Parameter alle möglichen Aktionen und die bisher ausgewählte Aktion erhält.

Quelltext 5.14: Ein kleiner Ausschnitt aus der Methode `Aktion getGueltigeAktion(...)`. Sie ist der letzte Schritt der Aktionsbestimmung im Spiellogikmodul.

```
1 if (bisherigeAktion == Aktion.FOLD
2   || bisherigeAktion == Aktion.CHECK) {
3   if (moeglAktionen.contains(Aktion.CHECK)) {
4     return Aktion.CHECK;
5   } else {
6     return Aktion.FOLD;
7   }
8 }
```

5.4 Modul 3: Interaktion

Das Interaktionsmodul beschränkt sich auf die Eingabe, einer zuvor berechneten Aktion, in die Benutzeroberfläche der Pokerplattform. Die Steuerung der Maus wird in Java durch die Klasse `java.awt.Robot` ermöglicht. Nachdem ein Objekt `robot` der Klasse `Robot` erzeugt wurde, ist das das Bewegen des Mauszeigers durch den Befehl `robot.mouseMove(X,Y)` möglich. Ein Mausklick erfolgt durch die Befehle `robot.mousePress()` und `robot.mouseRelease()`. Der Mauszeiger ist anschließend sofort auf der Position mit den Bildschirmkoordinaten (X,Y). Dieses Verhalten ist nicht erwünscht, da PokerBot möglichst menschenähnlich agieren soll. Aus diesem Grund wird eine Bewegung zwischen der aktuellen Mausposition und der Zielposition simuliert.

Simulation der Mausbewegung

Für die Simulation wurde eine Methode `mouseMove(int x, int y)` implementiert, welche die Membervariablen `Koordinate position` und `double speed` verwendet und den Mauszeiger schrittweise an die Zielposition (X,Y) bewegt. Die Methode prüft, ob die in `position` gespeicherte, aktuelle Mauszeigerposition links oder rechts von, beziehungsweise unter oder über der Zielposition ist. Anschließend wird der Mauszeiger um einige Pixel in die Richtung der Zielposition bewegt. Die Anzahl der Pixel, um die der Zeiger verschoben wird, entspricht dem in `speed` gespeicherten Wert. Ein Zufallswert verändert die Variable `speed` nach jedem Schritt um einen Wert zwischen -0.1 und $+0.1$, wobei darauf geachtet wird, dass der Wert niemals größer als sieben oder kleiner als zwei ist. Bei der Überschreitung der genannten Grenzen nimmt `speed` den Wert vier an. Quelltext 5.15 zeigt zwei Zeilen aus der Methode `mouseMove(...)` mit dieser einfachen, aber wirkungsvollen Lösung.

Quelltext 5.15:

```
1 speed = speed + (Math.random() * 0.2 - 0.1);
2 if (speed < 2 || speed > 7) speed = 4;
```

Die Zielposition bestimmen

Die Interaktion erfährt, an welchem Spiel, welche Schaltfläche zu klicken ist. Die Positionen der Schaltflächen errechnet das Interaktionsmodul aus der Position des Fensters, sowie der Position und Größe der jeweiligen Schaltfläche auf der entsprechenden Pokerplattform. Ähnlich wie bei dem Informationsgewinnungsmodul wird auch im Interaktionsmodul der Parameter `Spiel spiel` zur Übergabe dieser Informationen verwendet.

Kapitel 6

Auswertung

6.1 Aufwand der bildbasierten Softwareintegration

Durch den Ansatz der bildbasierten Softwareintegration konnten schnelle Fortschritte erzielt werden. Die Einarbeitungsphase war sehr kurz und nach wenigen Tagen wurden erste Elemente erkannt. Der Einsatz von Template Matching als primären Bildverarbeitungsalgorithmus hat sich als stabile und leicht zu handhabende Lösung bewährt. Damit das Verfahren allerdings gut funktioniert, ist ein gewisser Zeitaufwand für das Testen und Korrigieren von Parametern einzuplanen.

Die Komplexität der Benutzeroberfläche von Online-Poker-Plattformen hat sich als grenzwertig für den Einsatz der bildbasierten Softwareintegration herausgestellt. Es konnten zwar schnelle Erfolge erzielt werden, jedoch hat die vollständige Analyse der Benutzeroberfläche bis zum Ende Probleme bereitet.

6.2 Performance der Softwareintegration

Wie erwartet hat der bildbasierte Ansatz keine Performanceprobleme verursacht. Lediglich die Rate, mit welcher der Bildschirm abgetastet wird, hat sich als begrenzender Faktor gezeigt. Auf einem modernen Rechner sind Abtastraten von 5-10 Bildern pro Sekunde problemlos möglich. Für PokerBot wurden Abtastraten von etwa zwei Bildern pro Sekunde verwendet.

6.3 Qualität der bildbasierten Softwareintegration

Die Qualität der bildbasierten Softwareintegration hängt stark von der Qualität der zum Template Matching verwendeten Templates ab. Am besten lassen sich Aussagen über die Qualität, durch die Analyse der auftretenden Fehler, treffen.

6.3.1 Fehler beim Template Matching

Auch mit guten Templates müssen bei jedem neuen Einsatzgebiet die Parameter des Template Matching überprüft und gegebenenfalls angepasst werden. Bei einer Stichprobe aus 100 zufällig ausgewählten Händen, betrug die Fehlerquote bei der Erkennung in den meisten Bereichen 0 Prozent (siehe Tabelle 6.1).

In seltenen Fällen werden von einer Pokerplattform Werbung oder andere Informationen eingeblendet. Der Spieler muss diese eingeblendeten Informationen wegklicken, ansonsten verschwinden sie erst nach einigen Sekunden. In so einem Fall werden alle verdeckten Elemente nicht erkannt. Dieser Fall ist in der getesteten Stichprobe nicht aufgetreten.

Auswirkung der Fehler

Werden Spielelemente nicht erkannt, führt das im schlimmsten Fall zu einer Fehlentscheidung bei der aktuellen Hand. Sobald das Spielfenster wieder unverdeckt ist, kann das Spiel ohne Einbußen fortgesetzt werden.

Tabelle 6.1: Fehlerquote beim Template Matching. Alle Angaben sind in Prozent.

Stack	Karten	Buttons[1]	Dealer	Pot	Name [2]	Spieleranzahl
0	0	0	0	0	100	0

6.3.2 Fehler bei der Interpretation

Auch wenn das Template Matching die einzelnen Elemente richtig erkennt, kann es passieren, dass bei der Interpretation Fehler auftreten. Insbesondere bei der Erkennung der Aktionen von Mitspielern, kann es zu Problemen kommen. Tabelle 6.2 zeigt die Fehlerquote bei der Erkennung aller Daten, die nicht direkt aus dem Screnshot ausgelesen werden können. Bei der Stichprobe aus 100 Händen wurden besonders viele Hände, bei denen der Flop zu sehen war, ausgewählt. Hände, bei denen PokerBot sofort ausgestiegen ist, bieten wenig Fehlerpotential. Folgende Fehlerquellen sind bekannt:

- Fehler treten auf, wenn ein Spieler einen Einsatz macht, dann aber den Tisch verlässt.
- Dadurch, dass die Aktionserkennung auf der Analyse der Kontostände der Mitspieler basiert, kann es auch zu Fehlern kommen, wenn sich ein Kontostand innerhalb einer Runde plötzlich verändert.
- Eine weitere Fehlerquelle ist die Tatsache, dass nur ein Screenshot pro Wettrunde gemacht wird. Genau dann, wenn Pokerbot an der Reihe ist. Aus diesem Screenshot werden dann, anhand des Pots, der Kontoständer der Mitspieler und der sonstigen Elemente auf dem Bild,

Rückschlüsse auf den Spielverlauf gezogen. Es hat sich herausgestellt, dass es manchmal nicht trivial ist zu erkennen, welche Aktionen die anderen Spieler gemacht haben. Insbesondere, wenn viele Erhöhungen und Phasenwechsel stattgefunden haben.

Auswirkung der Fehler

Die genannten Fehler haben kaum spielerische Auswirkungen. In der großen Mehrheit aller Fälle spielt es keine Rolle, ob ein Mitspieler in der letzten Runde erhöht hat oder nur mitgegangen ist. Eine Beeinträchtigung ist die dadurch auftretende Inkonsistenz des Modells. Vergangene Spiele können in manchen Fällen nicht schrittweise rekonstruiert werden.

Tabelle 6.2: Fehlerkennungen bei der Interpretation.

	Gegneraktion[3]	SB\BB	SHPR[4]	Blatt[5]	Preflopblatt
Flop gesehen: 62	6	0	0	0 (2)	0
Nicht gesehen: 38	0	0	0	0 (0)	0

6.3.3 Fehler bei der Interaktion

In der gesamten Test- und Entwicklungsphase traten Fehler bei der Interaktion nur vereinzelt und unter Einfluss externer Kräfte auf. Es wird immer rechtzeitig auf die richtige Schaltfläche gedrückt. Einzig das Verhalten des Mauszeigers, beim Wechsel zwischen mehreren Bildschirmen, ist nicht optimal. Anstatt langsam zwischen den Bildschirmen hin und her zu wechseln, springt die Maus auf den nächsten Bildschirm und bewegt sich dann langsam zu der gewünschten Position.

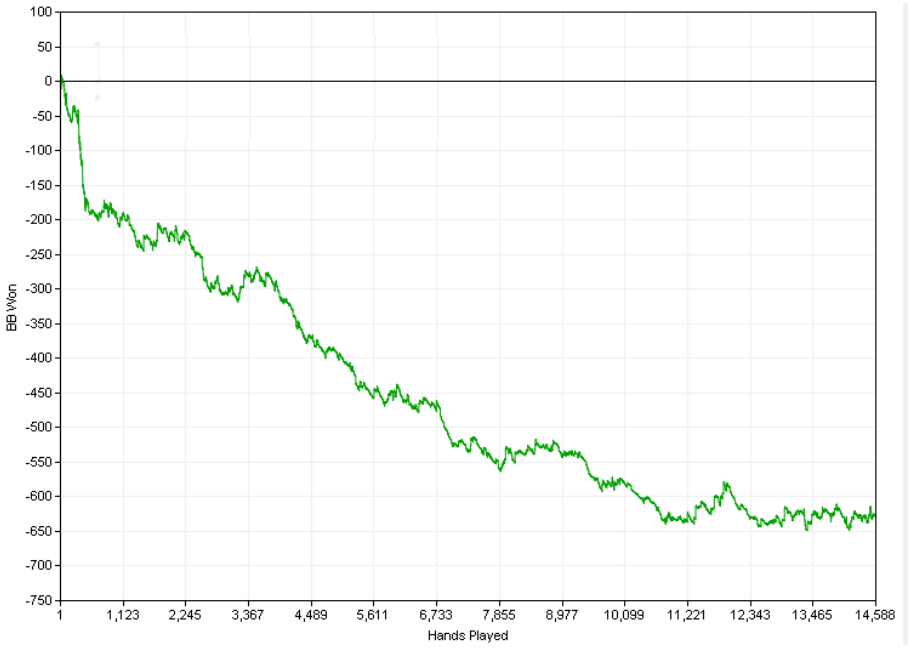
6.4 Spielerische Qualität von PokerBot

Der Schwerpunkt dieser Arbeit ist die bildbasierte Integration von Software. Die Evaluation der spielerischen Qualität eines Pokerprogramms erfordert optimalerweise eine gut angebundene Testumgebung und die Möglichkeit zur Durchführung vieler Tests. Diese Gegebenheiten sind alleine, weil PokerBot unter realen Bedinungen spielt, nicht gegeben. Fehler, die auftreten, sind häufig nicht unmittelbar auf die Künstliche Intelligenz zurückzuführen. Dennoch wird in diesem Abschnitt auf die spielerischen Qualitäten und Probleme von PokerBot eingegangen.

Kontinuierliche Verbesserung der KI

Zunächst wurde versucht, eine Pokeranleitung [25] für menschliche Spieler auf PokerBot zu übertragen. Nachdem alle Regeln vermeintlich implemen-

Abbildung 6.1: Spielerische Entwicklung von PokerBot.



tiert waren, hat PokerBot angefangen, online zu spielen. Durch die Beobachtung der Spielweise und den Einsatz der entwickelten Analysewerkzeuge wurden kontinuierlich Fehler gefunden und beseitigt. Insbesondere durch die Analyse der Hände, in denen PokerBot größere Verluste verzeichnet hat, konnten viele Fehler aufgedeckt werden. Es hat sich herausgestellt, dass die Zahl der Fälle, die durch die Regeln der Pokeranleitung nicht behandelt werden, größer ist, als erwartet. Das Abfangen aller Sonderfälle wurde mit zunehmender Komplexität immer problematischer.

Erst dank der stochastischen Überprüfung, kann auf die Behandlung vieler Sonderfälle verzichtet werden, da diese automatisch abgefangen werden.

Abbildung 6.1 visualisiert die spielerische Entwicklung von PokerBot. Der Graph zeigt die Anzahl gewonnener Big Blinds im Verhältnis zu den gespielten Händen. Nachdem in den ersten 11000 Händen ein deutlicher Abwärtstrend zu erkennen war, spielt PokerBot die letzten 3500 Hände ohne Verlust.

6.5 Ausblick

Sowohl in der KI als auch in der Informationsgewinnung gibt es noch Verbesserungsmöglichkeiten. Um die Einbindung neuer Pokerplattformen zu vereinfachen und zu beschleunigen, könnte ein weiteres Hilfsprogramm entwickelt werden. Dieses sollte in der Lage sein, den Prozess der Template-Erstellung

und deren Positionsbestimmung, weitgehend zu automatisieren. Die KI kann um eine Gegneranalyse erweitert werden, welche bereits gespielte Hände, der einzelnen Gegner, analysiert und auswertet. Des Weiteren könnte ein lernfähiger Algorithmus entwickelt werden, welcher die Parameter der mathematischen Überprüfung überwacht und selbstständig anpasst. Eine sehr wirksame, wenn auch nicht erlaubte, Funktion wäre die Kommunikation zwischen mehreren Instanzen von PokerBot. Hier könnten zum Beispiel die Handkarten der PokerBot's, welche am gleichen Tisch spielen, ausgetauscht und in die mathematische Überprüfung eingebunden werden.

Literaturverzeichnis

[1] B.B. Agarwal, S.P. Tayal, and M. Gupta. *Software engineering & testing: an introduction*. Infinity Science Series. Jones & Bartlett Publishers, Incorporated, 2009.

[2] LAB Asrpise! *User Manual: Asprise OCR SDK 4.0*. http://asprise.com/product/ocr/doc/DevelopersGuide-Java.pdf, 2007.

[3] Martin Bahr. *Veranstalten von Poker-Turnieren nicht strafbar*. http://glucksspiel-und-recht.de/urteile/Veranstalten-von-Poker-Turnieren-nicht-strafbar-Amtsgericht-Hamburg-20090107.html, January 2009.

[4] B. Blunden. *The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System*. Wordware Pub., 2009.

[5] R. Brunelli. *Template matching techniques in computer vision: theory and practice*. Wiley, 2009.

[6] Ron Cemer. Java ocr alpha version 1.101, http://sourceforge.net/projects/javaocr/.

[7] Elliot J. Chikofsky and James H. Cross. Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 7:13–17, 1990.

[8] Aaron Davodspm. Opponent modeling in poker: Learning and acting in a hostile and uncertain environment. Master's thesis, University of Alberta, 2002.

[9] P.Iglinski E. Stroulia, M. El-Ramly and P.Sorenson. User interface reverse engineering in support of interface migration on the web. In *Automated Software Engineering, Kluwer Academic Publishers*, 2003.

[10] Yousef El-Dardiry. *BuddyFuse*, volume C.Research Study. Bachelor Project, Department of Software Engineering (EEMCS) Delft University of Technology, 2008.

[11] Christopher Urmson et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25:425–466, Wiley InterScience, June 2008.

[12] Roland Fürst. Poker als risiko für die entstehung problematischen spielverhaltens bei jugendlichen. Master's thesis, Hochschule für Angewandte Wissenschaften Hamburg, 2009.

[13] Th. Haase, P. Klein, and M. Nagl. Software integration and framework development. In *Collaborative and Distributed Chemical Engineering. From Understanding to Substantial Design Process Support*, volume 4970 of *Lecture Notes in Computer Science*, pages 555–590. Springer Berlin / Heidelberg, 2008.

[14] Olaf Poneta Jan-Hendrik Borth. *PokerBot SVN*. https://ponetaborth.svn.sourceforge.net/svnroot/ponetaborth/Code/.

[15] W.S. Jawadekar. *Software Engineering: Principles and Practice*. Tata McGraw-Hill, 2004.

[16] Lanyan Kong, Eleni Stroulia, and Bruce Matichuk. Legacy interface migration: A task-centered approach. In *Proceedings of HCI International (the 8th International Conference on Human-Computer Interaction) on Human-Computer Interaction: Ergonomics and User Interfaces-Volume I*, pages 1167–1171. L. Erlbaum Associates Inc., 1999.

[17] Karl Kurbel. *Entwicklung und Einsatz von Expertensystemen*. Springer, 1989.

[18] Torbjörn Lofterud. *Poker bots*. Lecture at Chaos Communication Camp, http://events.ccc.de/camp/2011/Fahrplan/track/Science/4424.en.html, August 2011.

[19] Jan Meinert. *Die Pokerschule*. Knaur, 2007.

[20] B. Neumann. *Bildverarbeitung für Einsteiger: Programmbeispiele mit Mathcad*. Springer, 2004.

[21] Denis Richard Papp. Dealing with imperfect information in poker. Master's thesis, University of Alberta, 1998.

[22] Cyrus Peikari and Anton Chuvakin. *Security warrior - know your enemy*. O'Reilly, 2004.

[23] Mike Perry and Nasko Oskov. *Introduction to Reverse Engineering Software*. Free Tech Books, http://www.savs.hcc.edu.tw/~chuavv/articles/RevEng/index.html, 2004.

[24] Pokerstrategy. *Welche Pokervariante liegt dir am besten?*, volume Welche Vorteile Bietet Fixed Limit? Article, http://de.pokerstrategy.com/strategy/others/1271/1/, 2008.

[25] Pokerstrategy. *Fixed Limit Basic Handout*, volume 02042008-PS-FL-BASIC-HANDOUT-DE-V01. PokerStrategy, http://resources.pokerstrategy.com/Strategy/pdf/ps_fl_basic_handout_de.pdf, 2009.

[26] Ilya Mezhirov Ray Smith, Thomas Breuel. Tesseract-ocr, http://code.google.com/p/tesseract-ocr/.

[27] Hofer Reinhard. *Onlinepoker Texas Hold ´em: Die letzten Worte eines Pokerspielers...* Books on Demand GmbH, Norderstedt, 2010.

[28] Reuven Y Rubinstein. *Simulation and the Monte Carlo method*. Wiley, 1981.

[29] Terence Schauenberg. Opponent modeling and search in poker. Master's thesis, University of Alberta, 2006.

[30] Shanky Technologies. Holdem bot user manual, http://www.bonusbots.com/HoldemBotUserManual.pdf.

[31] Shanky Technologies. Holdem poker bot, http://www.bonusbots.com/holdempokerbot.htm.

[32] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. Sikuli: using gui screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 183–192. ACM, 2009.

[33] C Zhang and Zhengyou Zhang. *A Survey of Recent Advances in Face Detection*. Microsoft Research Technical Report, MSR-TR-2010-66, June 2010.