Entwicklung eines Vier-Gewinnt Roboters

Hannes Schreiber, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Paper soll die Entwicklung eines selbstständig spielenden Vier-Gewinnt Roboters beschrieben werden. Ausgehend von der mathematischen Spieltheorie wird ein Algorithmus entwickelt, mit dem der Roboter intelligent spielen soll. Anschließend wird die Realisierung des Einwurfmechanismus sowie der Erkennung der Spielsteine mittels Webcam erläutert. Die Funktionalität des Roboters wird an einem menschlichen Gegenspieler getestet.**

***Schlagwörter*—Künstliche Intelligenz, Lego Mindstorms, Projektseminar, Spieltheorie, Vier Gewinnt**

I. Einleitung

KÜNSTLICHE Intelligenz spielt heutzutage eine zentrale Rolle bei Computerspielen. Häufig wird nicht gegen menschliche Gegner, sonder gegen Computergegner gespielt. Dabei ist es für den eigenen Spielspaß und für ein realistisches Spielgefühl besonders wichtig, dass der künstliche Gegner ein möglichst menschliches Spielverhalten aufweist. Einen Bereich von Computerspielen bilden dabei Gesellschaftsspiele, die gegen einen Computergegner gespielt werden. Bei solchen Spielen kommt häufig die mathematische Spieltheorie zum Einsatz, um die künstliche Intelligenz zu entwickeln. Beispiele für solche Computerspiele sind Schach, Poker oder auch das in diesem Paper behandelte Vier-Gewinnt. Für gewöhnlich findet dabei das Spielerlebnis lediglich auf dem Bildschirm statt. In unserem Projekt sollte die Spielerfahrung dahingehend erweitert werden, dass wie beim echten Spiel auf einem realen Spielfeld mit physischen Spielsteinen gespielt werden kann. Dazu wurde ein Roboter entwickelt, der mittels einer Webcam die aktuelle Position aller Spielsteine auf dem Feld erkennt. Anschließend wird der nächste Spielzug berechnet, den der Roboter dann selbstständig ausführt, indem er zur jeweiligen Position fährt und seinen Stein einwirft.

II. Vorbetrachtungen

Zur Entwicklung einer künstlichen Intelligenz wurden Ansätze aus der mathematischen Spieltheorie genutzt. Dazu muss das behandelte Spiel zunächst genauer beschrieben werden.

A. Das Spiel Vier-Gewinnt

Bei Vier-Gewinnt spielen zwei Spieler gegeneinander und versuchen, 4 Steine ihrer Farbe horizontal, vertikal oder diagonal in eine Reihe zu bekommen. In Abbildung 1 ist ein Spielfeld mit einer Spielsituation, in der der Spieler mit den roten Steinen gewinnen würde, dargestellt. Es handelt sich um ein endliches Zwei-Personen-Nullsummenspiel mit perfekter Information. Endlich bedeutet, dass es einen Maximalwert an

DOI: 10.24352/UB.OVGU-2018-060

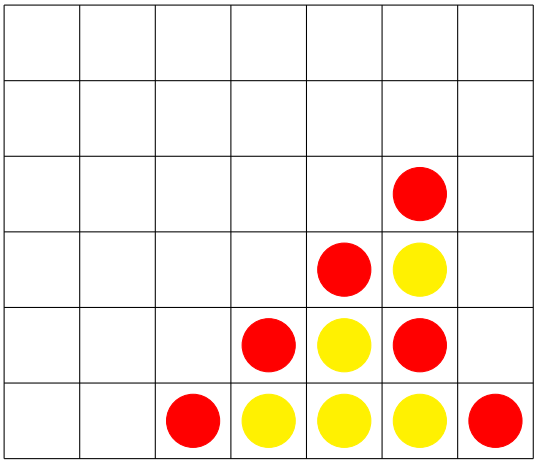

Abbildung 1. Vier-Gewinnt Spielfeld mit einer beispielhaften Spielsituation

Zügen gibt, nach denen das Spiel beendet ist. Nullsummenspiel beschreibt ein Spiel, bei dem die Summe der Gewinne und Verluste beider Spieler zusammen null ist. Ein Gewinn für den einen Spieler bedeutet also den gleichen Verlust für den anderen Spieler. Bei einem Spiel mit perfekter Information ist die komplette Spielsituation jederzeit für alle Spieler einsehbar (beispielsweise im Gegensatz zu Poker).

B. Bewertung einer Spielsituation

Zur Nutzung von Algorithmen aus der Spieltheorie ist es nötig, verschiedene Spielsituationen zu bewerten. Sei $\mathbf{A}$ eine 6x7 Matrix, die das Spielfeld mit den aktuellen Spielsituation repräsentiert. Für die Situation aus Abbildung 1 würde die Matrix beispielsweise folgendermaßen aussehen:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 & 1 & 1 & -1 \end{bmatrix}$$

Welche Zahl welcher Spielfarbe zugeordnet wird, ist dabei zunächst frei wählbar, hier wurde für die roten Steine die Zahl -1, für die Gelben die Zahl 1 und für ein leeres Feld die Zahl 0 gewählt. Eine ideale Bewertungsfunktion f

$$g = f(\mathbf{A}) = \begin{cases} 1, & \text{Spieler 1 gewinnt} \\ -1, & \text{Spieler 2 gewinnt} \\ 0, & \text{Unentschieden} \end{cases} \quad (1)$$

würde einer Spielsituation $\mathbf{A}$ den Wert g zuordnen. Damit der Algorithmus Züge die möglichst schnell zum Sieg führen bevorzugt, wird die Bewertung um einen Faktor erweitert.

$$g = f(\mathbf{A}) \cdot (43 - n) \tag{2}$$

Dabei gibt n die Nummer des jeweiligen Zuges an (von 1 bis maximal 42). Je früher ein möglicher Sieg im Spiel auftritt, desto höher wird dieser also bewertet. In der Praxis ist es jedoch häufig zu aufwändig, alle möglichen Spielsituationen zu berechnen und zu bewerten. Daher werden von der aktuellen Spielsituation ausgehend die möglichen Spielsituationen nur für eine bestimmte Anzahl an Zügen betrachtet. Eine mögliche Erweiterung besteht jetzt noch darin, ein Unentschieden genauer zu bewerten, also abzuschätzen, ob die jeweilige Spielsituation eine gute oder schlechte Ausgangssituation für die weiteren Züge darstellt. Somit wird auch einem Unentschieden ein Wert ungleich 0 zugeordnet und f hat mehr als 3 mögliche Ergebnisse.

C. MinMax-Algorithmus

Wenn die möglichen Spielsituationen einmal bewertet sind, kann der MinMax-Algorithmus genutzt werden, um den besten aktuell möglichen Zug zu ermitteln. Der MinMax-

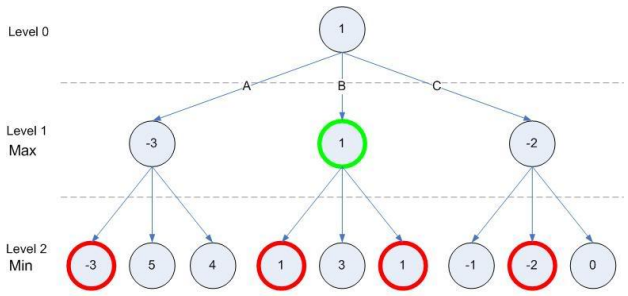

Abbildung 2. Beispielhafter Suchbaum für den MinMax-Algorithmus

Algorithmus basiert auf der Annahme, dass jeder Spieler stets versucht, seinen Gewinn zu maximieren bzw. seinen Verlust zu minimieren. Wenn der Spieler, der gerade an der Reihe ist, seinen Zug plant, kann er also davon ausgehen, dass er selbst stets die beste Situation, also die mit dem höchsten Wert für g, wählt. Er selbst will ja seinen Gewinn maximieren. Der Gegner wird stets die Situation mit dem niedrigsten Wert für g wählen, da er ja seinen Verlust minimieren will. Das führt dazu, dass in dem Suchbaum stets abwechselnd der niedrigste und der höchste Wert gewählt wird. Ein solcher Suchbaum ist in Abbildung 2 beispielhaft dargestellt. Es ist dazu zu sagen, dass aus Platzgründen nur ein Suchbaum mit einer Breite von 3 dargestellt ist. Das bedeutet, dass es bei jedem Zug 3 Möglichkeiten gibt. Bei Vier Gewinnt wäre es ein Suchbaum mit einer Breite von 7, jedoch ändert das nichts am Algorithmus selbst. Die Suchtiefe beträgt in diesem Beispiel den Wert 2. Man sieht, dass für jeden Zweig zunächst der minimale Wert der darunter liegenden Möglichkeiten übernommen wird. Anschließend wird für die Ausgangssituation der maximale Wert der 3 darunter liegenden Möglichkeiten übernommen. In diesem Beispiel würde der Algorithmus also den Zug B als beste Möglichkeit auswählen. [1]

III. Hauptteil

Der Roboter sollte also nicht nur ausgehend von der aktuellen Spielsituation einen Spielzug wählen, sondern auch die Steine selbstständig einwerfen. Dazu waren verschiedene Arbeitsschritte nötig.

A. Das Anfahren einer Spielposition

Damit der Roboter eine bestimmte Spielposition anfahren kann, muss er sich horizontal entlang des Spielfeldes bewegen können. Dazu wurde der Roboter mit vier Rädern ausgestattet, welche über einen Motor angetrieben wurden. Damit ein einwerfen von oben möglich war, wurde aus Kartons eine Plattform gebaut, auf welcher der Roboter fahren konnte. Es war von großer Wichtigkeit, dass die Bewegung des Roboters parallel zum Spielfeld erfolgt, da sich ansonsten der Abstand des Roboters zu den Schlitzen, in die die Spielsteine geworfen werden müssen, verändert, sodass das Einwerfen der Spielsteine nicht mehr funktioniert. Es hat sich gezeigt, dass trotz großer Sorgfalt kein exaktes paralleles Fahren realisiert werden konnte. Da sich beim Hin- und Herfahren die Fehler gegenseitig ungefähr aufheben, konnte trotzdem ein sicheres Einwerfen der Spielsteine realisiert werden. Es war jedoch wichtig, den Roboter zu Beginn des Spiels möglichst parallel zum Spielfeld hinzustellen. Im Programm wurde das Anfahren so realisiert, dass eine Funktion geschrieben wurde, die als Parameter die aktuelle Position und die vom Algorithmus berechnete, anzufahrende Position übergeben bekommt und daraus die Strecke ermittelt, die gefahren werden muss.

B. Das Einwerfen der Spielsteine

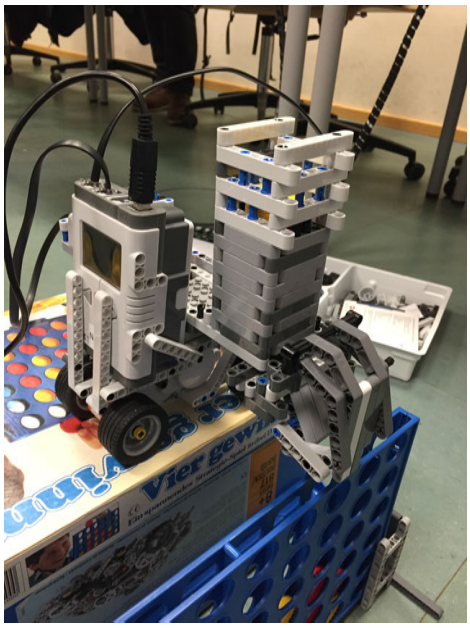

Abbildung 3. Das Spielsteinlager

Ist der Roboter zu der gewünschten Spielsituation gefahren, muss er einen Spielstein in den vorgesehenen Schlitz des Spielfeldes werfen. Damit der Roboter ein gesamtes Spiel ohne menschliches Eingreifen spielen kann, wurde ein Lager gebaut,

das Platz für die 21 Spielsteine hat. In Abbildung 3 ist das Spielsteinlager zu sehen. Aus diesem Lager sollte nun immer ein Stein eingeworfen werden. Das erwies sich als besonders schwierig, da die Spielsteine, da sie natürlich nicht für solch eine Nutzung gedacht sind, kleine Einkerbungen haben, damit sie sich ineinander verhaken. Somit konnte der unterste Stein im Lager nicht einfach herausgeschoben werden, da er durch die Einkerbungen und den Druck der darüber liegenden Steine verhakt war. Als Lösung wurde ein Schieber gewählt, der,

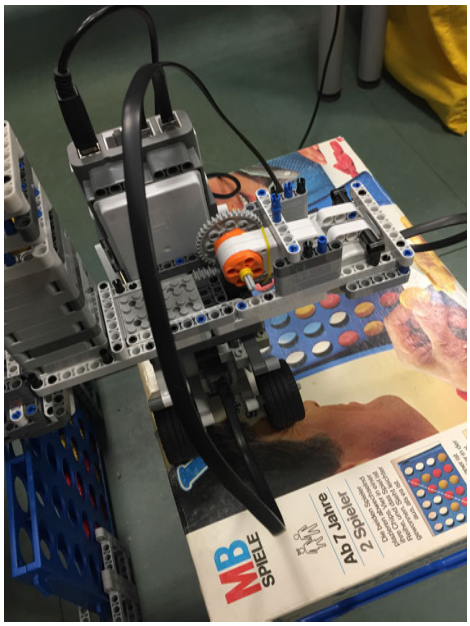

Abbildung 4. Die Abwurfvorrichtung

angetrieben durch einen Motor, einen sehr ruckartigen, starken Stoß ausführt. Problematisch war hierbei, dass die nötige Kraft für das Herausschießen des untersten Steines mit steigender Spieldauer sinkt, da der Druck auf den Stein kleiner wird, wenn weniger Spielsteine über ihm liegen. Der Motor für den Schieber musste also eine Drehmoment aufwenden, dass groß genug war, um den untersten Stein herauszuschießen. Jedoch wurden die Spielsteine so eben mit sinkender Anzahl der Reststeine im Lager immer weiter herausgeschossen. Um trotzdem bei jedem Zug den Schlitz zu treffen, wurde eine Vorrichtung gebaut, die die Flugbahn der Spielsteine so umlenkt, dass sie immer im Schlitz endet. Die Abwurfvorrichtung ist in Abbildung 4 dargestellt.

C. Das Erkennen der aktuellen Spielsituation

Das Erkennen der Spielsteine sollte zunächst durch einen Farbsensor realisiert werden. Dazu sollte der Sensor an einem Arm befestigt werden, der sich, angetrieben durch einen Motor, vertikal entlang des Spielfeldes bewegt und so die Position der Spielsteine und deren Farbe erkennt. Die Idee war dabei, dass sich das Programm die vorherige Spielsituation merkt, sodass nicht alle Felder, sonder nur die Felder, in denen ein Stein liegen könnte, abgefahren werden. Im Laufe des Seminars wurde jedoch entschieden, dass das Erkennen der Spielsituation mit Hilfe einer Webcam realisiert werden soll. Das brachte eine Erleichterung im mechanischen Teil des Projekt mit sich, da keine Vorrichtung zum Abfahren der Felder gebaut werden musste. Andererseits brachte es zusätzliche Arbeit im Bereich der Programmierung, da zusätzlich eine Bildverarbeitung durchgeführt werden musste. Für die Kalibrierung der Kamera wurde ein recht einfacher Weg gewählt. Für den Nutzer wurde ein Kamerabild mit markierten Bildpunkten angezeigt. Zur Kalibrierung musste der Nutzer nun die Webcam so ausrichten, dass die Markierungen genau in den Feldern lagen. Im Programm wurden nun die Helligkeitswerte an den einzelnen Markierungen überprüft und daraus konnte entschieden werden, ob es sich um einen roten oder gelben Spielstein oder um ein leeres Feld handelt. Zur besseren Erkennung des leeren Felder wurde ein schwarzes Blatt hinter das Spielfeld geklebt.

D. Das Finden des besten Zuges

Aus Zeitgründen wurde ein eher einfacher Algorithmus gewählt. Der Algorithmus erkennt, wenn er eine Gewinnsituation hat und nutzt diese, außerdem erkennt er die Gewinnsituation des Gegners und verhindert diese. Sollte nichts davon der Fall sein, wird versucht, den eigenen Stein an eine bereits bestehende Reihe anzureihen, es wird also versucht, eine Gewinnsituation aufzubauen. Das entspricht einem MinMax-Algorithmus mit einer Suchtiefe von 2.

E. Das gesamte Programm

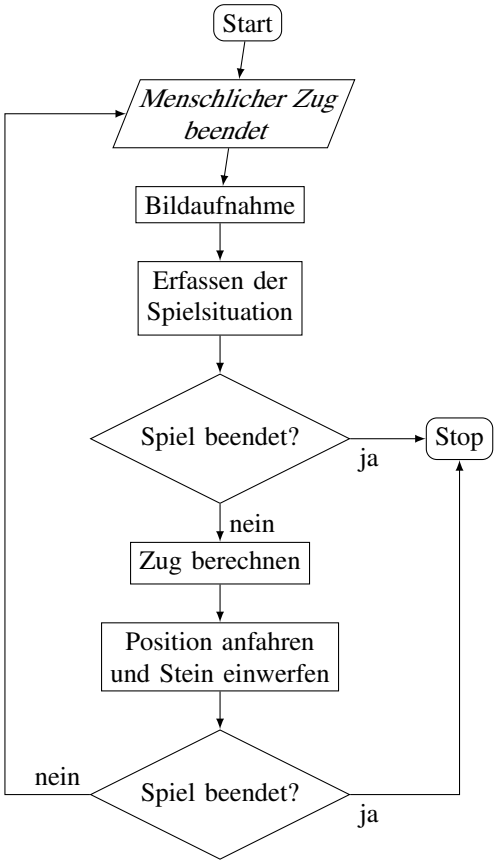

Abbildung 5. Programmablaufplan für das Spielprogramm

In Abbildung 5 ist ein Programmablaufplan für das gesamte Programm dargestellt. Nachdem die Kamera kalibriert wurde

wird das Spiel gestartet. Der Gegner muss über eine Benutzeroberfläche signalisieren, wenn er seinen Zug beendet hat. Das Programm wartet solange auf die Eingabe 'Menschlicher Zug beendet'. Anschließend wird mittels Webcam die aktuelle Spielsituation erfasst. Danach wird geprüft, ob der Gegner bereits gewonnen hat. Falls das der Fall ist, stoppt das Programm. Falls das nicht der Fall ist, berechnet das Programm den bestmöglichen Zug, fährt zu dieser Position und wirft einen Stein ein. Anschließend wird geprüft, ob der Roboter gewonnen hat. Sollte das der Fall sein, wird das Programm gestoppt. Ist das nicht der Fall, wird wieder gewartet bis der menschliche Spieler signalisiert, dass er seinen Zug beendet hat.

IV. Ergebnisdiskussion

Am Ende des Seminars war ein Roboter entwickelt worden, der gegen einen menschlichen Spieler selbstständig spielen konnte. In einem Testspiel schaffte es der Roboter, die Testperson zu besiegen, wobei die Testperson allerdings absichtlich Gewinnchancen des Roboters nicht verhinderte. Es zeigte sich, dass der Roboter erfolgreich Gewinnsituationen des Gegners verhinderte und eigene Gewinnsituationen kreierte und diese dann auch nutzte. Große Probleme bereitete zunächst die Konstruktion der Einwurfvorrichtung. Es erwies sich aufgrund der in Unterabschnitt III-B beschriebenen ungleichmäßigen Flugweiten der Spielsteine als schwierig, eine Einwurfvorrichtung zu entwickeln, die unabhängig von der Anzahl der bereits getätigten Züge immer den Schlitz trifft. Dazu wurde eine Schiene konstruiert, die die Flugbahn der Steine lenken sollte. Letztendlich konnte so eine sehr hohe Trefferquote erreicht werden. Ein weiteres Problem stellte das Spielsteinlager dar. Dieses wurde zuerst mit einigen Öffnungen an der Seite konstruiert. Dies stellte sich jedoch später alsm Fehlerquelle heraus, da die Steine in dieses Öffnungen häufig stecken blieben. Deshalb wurde das Lager so verändert, dass es komplett geschlossen war. Das brachte den Nachteil mit sich, dass das Befüllen des Lagers sehr lange dauerte und dass bei verkanteten Steinen das Lager abgebaut werden musste. Auch die Erkennung der Spielsituation mittels Webcam stellte ein Problem dar. Die Kalibrierung der Kamera dauerte immer sehr lang, da die markierten Bildpunkte genau in den einzelnen Feldern liegen mussten. Außerdem musste die Kamera, nachdem sie selbst oder das Spielfeld auch nur ein kleines bisschen berührt wurde, neu kalibriert werden. Außerdem funktionierte die Erkennung der Farben der Spielsteine nur bei bestimmten Lichtverhältnissen. Änderten sich die Lichtverhältnisse, konnte die Bildverarbeitung nicht mehr zwischen roten und gelben Spielsteinen unterscheiden. Aufgrund dieser Probleme blieb relativ wenig Zeit für den eigentlichen Algorithmus, der den besten Zug ermittelt. Daher ist dieser leider ziemlich einfach gehalten, sodass der Roboter nicht wirklich gut spielen konnte.

V. Zusammenfassung und Fazit

In dem LEGO Mindstorms Seminar wurde erfolgreich ein Roboter entwickelt, der selbstständig Vier Gewinnt gegen einen menschlichen Gegner spielt. Der Roboter könnte auf verschieden Weisen verbessert werden. Zum Einen könnte die Kalibrierung der Webcam etwas automatisiert werden. Ein Vorschlag eines Betreuers wäre, die Vier Eckpunkte des Spielfeldes durch den Nutzer im Kamerabild markieren zu lassen. Aus diesen vier Punkten könnte dann die Lage der einzelnen Felder berechnet werden. So wäre eine schnelle Kalibrierung unabhängig vom Abstand und Winkel der Kamera zum Spielfeld möglich. Außerdem könnte die Kamera zusätzlich auf die Lichtverhältnisse kalibriert werden. So könnte man bei der Kalibrierung zwei unterschiedlich farbige Steine in zwei festgelegten Felder platzieren. Das Programm könnte dann die Helligkeitswerte dieser Steine ermitteln und die Werte dann, mit einer gewissen Toleranz, als Grenzen für einen roten bzw. gelben Stein nutzen. Desweiteren könnte das Anfahren der Positionen so verbessert werden, dass der Roboter immer parallel zum Spielfeld fährt. Dazu könnte eine Schiene gebaut werden, die parallel zum Spielfeld verläuft und verhindert, dass der Roboter von seinem Kurs abkommt. Außerdem könnte das Spielsteinlager so verändert werden, dass es nicht mehr gerade nach oben sondern schräg verläuft. So würde der Druck auf den untersten Stein durch die darüber liegenden Steine stark verringert werden. Desweiteren könnte des Algorithmus, der den besten Zug ermittelt, verbessert werden. Mit einer richtigen Implementierung des MinMax-Algorithmus wäre es möglich, einen Roboter zu erschaffen, der weitaus besser spielt. Es wäre denkbar, dass die Suchtiefe über eine Benutzeroberfläche verändert werden kann, sodass man einstellen kann, wie gut der Roboter spielt. Bei einer ausreichenden Suchtiefe wäre sogar ein perfekt spielender Roboter denkbar. Außerdem könnte man einbauen, dass der Roboter einen ungültigen Zug des Gegners erkennt. Über den Vergleich der vorherigen Anzahl der Spielsteine und der aktuellen würde der Roboter es erkennen können, wenn der Gegner keinen oder zwei Steine eingeworfen hat.

Literaturverzeichnis

[1] Wikipedia, *Minimax-Algorithmus*, https://de.wikipedia.org/wiki/Minimax-Algorithmus (Abruf: 10.03.2018)

[2] H. Baier, *Der Alpha-Beta-Algorithmus und Erweiterungen bei Vier Gewinnt*, 2006