Lego Mindstorms Vier-Gewinnt-Roboter

Robin Henryk Drews, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In diesem Paper geht es um die Entwicklung und Konstruktion eines autonom spielenden Vier-Gewinnt-Roboters mit Lego Mindstorms. Dazu wird der MiniMax-Algorithmus und die Spielerkennung per Webcam verwendet. Der implementierte Algorithmus basiert auf der Spieltheorie. Dies soll dem Roboter ermöglichen intelligente Spielzüge zu machen. Auch die Realisierung des Roboters wird in diesem Paper verdeutlicht, in dem die Konstruktion und der Einwurfmechanismus erklärt werden.**

***Schlagwörter*—Lego Mindstorms, Vier-Gewinnt, Projektseminar, Spieltheorie, MiniMax-Algorithmus, Robotik.**

I. Einleitung

HEUTZUTAGE sind künstliche Intelligenzen ein wichtiges Thema, sie findet in vielen Bereichen unseres Lebens Anwendungsgebiete, z.B. in der Diagnostik, bei der Gesichts- oder Bilderkennung und in Computerspielen. Besonders in Computerspielen hat sie eine zentrale Rolle eingenommen. Somit ist es nicht mehr zwingend notwendig, einen anderen menschlichen Gegenspieler zu haben. Jedoch braucht die künstliche Intelligenz dabei ein an Menschen orientiertes Spielverhalten, um ein realistisches Spielgefühl zu erzeugen. Dieses Spielverhalten ist besonders bei Gesellschaftsspielen relevant. Um eine künstliche Intelligenz für diese Spiele zu entwickeln, kommt oftmals die mathematische Spieltheorie zum Einsatz. In Spielen wie Skat, Dame oder auch 4-Gewinnt findet das Spielgeschehen nur auf dem Bildschirm statt. Um dazu eine Alternative zu schaffen wird in diesem Paper ein Roboter behandelt, welcher an einem physischen Spielfeld 4-Gewinnt spielen kann. Dabei wird das Spielfeld mittels einer Webcam erkannt und dann ausgewertet. Der Roboter berechnet darauf beruhend seinen eigenen Spielzug und wirft den Spielstein autonom ein.

II. Vorbetrachtungen

Um eine künstliche Intelligenz für den Roboter zu entwickeln muss das Spiel zunächst genauer beschrieben werden. Danach können Ansätze aus der Spieltheorie angewandt werden.

A. 4-Gewinnt

4-Gewinnt ist ein zwei Personen Gesellschaftsspiel, bei dem es das Ziel ist, 4 der eigenen Steine diagonal, horizontal oder vertikal in einer Reihe zu platzieren. Ein Beispiel für eine Gewinnsituation ist in der Abbildung [1] dargestellt. Um eine solche oder ähnliche Situation zu erreichen hat jeder Spieler 21 Spielsteine. Wie auch Dame und Schach ist 4-Gewinnt ein endliches Nullsummenspiel mit perfekter Information. Die Bezeichnung „endlich“ gibt hierbei nur an, dass es eine maximale Anzahl von Zügen gibt. Mit dem Ausdruck „Nullsummenspiel“ beschreibt man ein Spiel, welches als Summe der Verluste und Siege beider Gegenspieler Null hat. Bei einem Spiel mit perfekter Information können beide Spieler alle vorherigen Spielzüge nachvollziehen und haben immer einen aktuellen Überblick über das Spielgeschehen. Ein Beispiel für Spiele ohne perfekter Information sind Kartenspiele.

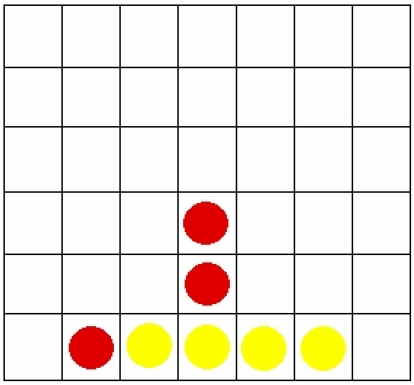

Abbildung 1. Spielfeld mit Gewinnsituation für Gelb, Quelle: o.S. aus http://www.mathematik.uni-muenchen.de/~spielth/artikel/VierGewinnt.pdf Diagramm I1.1

B. Erfassung und Auswertung der Spielsituation

Um die Spielsituation zu erkennen wird das Spielfeld in eine 6x7 Matrix vereinfacht. Eingeworfene Steine und leere Felder bekommen dann einen Wert zugeteilt. Für das Beispiel in Abbildung [1] hieße die Matrix M. Somit hätte diese Matrix die folgende Form:

$$\mathrm{M} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 2 & 2 & 2 & 0 \end{bmatrix}$$

Dabei bekommt Rot die Zahl 1 und Gelb die Zahl 2 zugeteilt, jedes leere Feld hat die Zahl 0. Somit wird bei einer Spielsituation M erkannt, wo welche Spielsteine liegen. Nun lässt sich eine Bewertungsfunktion, ähnlich der Funktion [1] aufstellen, welche die Gewinnchancen für den jeweiligen Spieler berechnet.

$$f(M) = \begin{cases} 0, & \text{Unentschieden} \\ +1, & \text{Spieler gewinnt} \\ -1, & \text{Gegenspieler gewinnt .} \end{cases} \quad (1)$$

10.24352/UB.OVGU-2019-039

Diese ist jedoch hauptsächlich für simplere Spiele, wie Tik-Tak-Toe, geeignet. Für komplexere Spiele wie 4-Gewinnt müsste diese Bewertungsfunktion noch angepasst werden: [2].

$$f(M) = \begin{cases} \infty & \text{Spieler gewinnt} \\ n \in \mathbb{N} > 0 & \text{Vorteil für Spieler} \\ 0 & \text{kein Vorteil für beide Spieler} \\ n \in \mathbb{N} < 0 & \text{Vorteil für Gegenspieler} \\ -\infty & \text{Gegenspieler gewinnt} \end{cases} \quad (2)$$

Dann kann mithilfe eines Algorithmus der nächste Spielzug berechnet werden. Dieser sucht nach Zügen mit dem höchsten Zahlenwert. Jedoch sollen die Züge, die möglichst schnell zum Sieg führen, bevorzugt werden.

C. MiniMax-Algorithmus

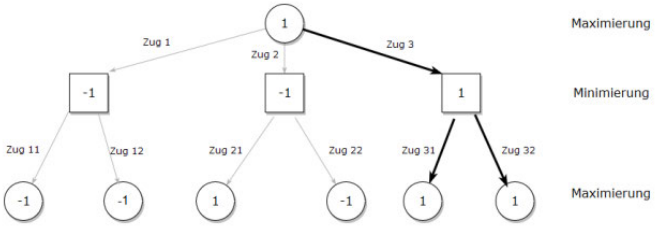

Abbildung 2. Beispielhafter Suchbaum beim MiniMax-Algorithmus, Quelle: http://www.gm.fh-koeln.de/ciopwebpub/Galitzki17.d/Galitz17.pdf S.16 Abbildung 5

Für das Herausfinden der besten Spielstrategie bei 4-Gewinnt eignet sich der MiniMax-Algorithmus. Dieser ist für endliche Zwei-Personen-Nullsummenspiele mit perfekter Information ausgelegt. Die Annahme auf der dieser Algorithmus basiert ist, dass jeder Spieler stets seine Gewinnchance maximieren will. Somit soll er die maximale Gewinnchance für den Spieler berechnen. Hierbei werden Suchbäume verwendet um die möglichen Züge zu simulieren. In Abbildung [2] ist ein Suchbaum dargestellt. Zur Veranschaulichung hat dieser jedoch nur eine Breite von Drei und eine Suchtiefe von 2. Ein Suchbaum für 4-Gewinnt bräuchte eine Breite von Sieben. Der Ausgangszustand des Suchbaumes ist ein Wurzelknoten. Von diesem aus werden alle Zugmöglichkeiten und die daraus folgenden Zugmöglichkeiten berechnet. Züge mit einer höheren Gewinnchance werden mit einem höheren Zahlenwert beschrieben. Der Zahlenwert ergibt sich daraus, dass der Spieler in seinem Zug das Maximum der Werte wählt. Bei den Knoten, wo der Gegenspieler am Zug ist, wird immer das Minimum der Werte genommen. Dies ist dadurch bedingt, dass der Spieler seinen Zug stets frei wählen kann. Der Zug des Gegenspielers kann jedoch nicht kontrolliert werden. Züge mit höheren Werten werden von der künstlichen Intelligenz bevorzugt. Nun wird immer abwechselnd der „Max- und Min-Zug“ simuliert, bis das Spiel zu Ende ist. Jedoch stellt dies bei komplexeren Spielen wie Schach und 4-Gewinnt ein Problem dar. Die Suchbäume sind zu groß und benötigen Unmengen an Speicher und Leistung. Im Beispiel von Abbildung [2] wäre der Zug 3 der vom Algorithmus bevorzugte Spielzug, da er den höchsten Zahlenwert darstellt. Es ist auch gut zu erkennen, dass immer abwechselnd ein „Max- und Min-Zug“ simuliert wird. Somit würde die Gewinnchance der künstlichen Intelligenz gesteigert werden und das Spielerlebnis würde authentischer werden.

III. Hauptteil

A. Spielfelderkennung

Zur Erkennung des Spielfeldes gab es mehrere Ansätze. Zunächst wurde versucht das Spielfeld mittels eines Farbsensors in Matlab umzusetzen. Jedoch erwies sich dies als äußert umständlich bei der mechanischen Umsetzung, weshalb dieser Ansatz verworfen wurde. Der zweite Ansatz war die Verwendung einer Webcam. Hierbei wurden die Ecken des Spielfeldes auf dem Bild festgelegt. Auf diesen beruhend wurden dann die Positionen der restlichen Zellen berechnet und festgelegt. Dies stellte sich als Zuverlässig heraus.

B. Berechnung des besten Spielzuges

Aufgrund der begrenzten Zeit wurde für die Berechnung der Spielzüge ein eher einfacher Algorithmus geschrieben. Dieser erkannte, ob er die eigene Reihe vervollständigen konnte. War dies der Fall, wurde dieser Zug priorisiert. Wenn jedoch der Gegenspieler seine Reihe vervollständigen konnte, versuchte der Algorithmus dies zu blockieren. Trat keiner der beiden Fälle ein, warf die künstliche Intelligenz an einer zufälligen Stelle einen Spielstein ein. Hierzu war es notwendig, dass die Spielsituation erfasst wurde, wie in [II-B] beschrieben.

$$f(M) = \begin{cases} 0, & \text{kein Ergebnis} \\ 1, & \text{Rot gewinnt} \\ 2, & \text{Gelb gewinnt} \\ 3, & \text{Unentschieden} \\ 4, & \text{Betrug} \end{cases} \quad (3)$$

Danach wertete eine Prüfroutine aus, ob ein Gewinner feststeht, es unentschieden ist oder betrogen wurde. Eine solche Routine ist mit Funktion [3] dargestellt. Dabei wurde als Betrug das Einwerfen von Zwei Spielsteinen oder das Einwerfen von keinem Spielstein angesehen. Bei dem Kriterium Null wurde der oben genannte Algorithmus ausgeführt.

C. Mechanische Umsetzung des Roboters

Um der künstlichen Intelligenz das 4-Gewinnt spielen zu ermöglichen, wurde aus dem Lego-Mindstorms-Set ein Roboter gebaut. Dieser verwendet einen Motor als Antrieb, um die einzelnen Zellen des Spielfeldes anzufahren. Dabei fährt er immer parallel zum Spielfeld. Die Distanz zwischen den Zellen wurde ausgemessen und als fester Wert im Programm eingetragen. Hierbei war es wichtig, zu testen ob der erste Stein in der Ausgangsposition die Zelle genau trifft. Beim Einwerfen der Spielsteine gab es anfangs das Problem, dass das Spielfeld zu hoch war. Dieser Höhenunterschied wurde durch den Bau eines Turmes überwunden. Auf diesem wurde das

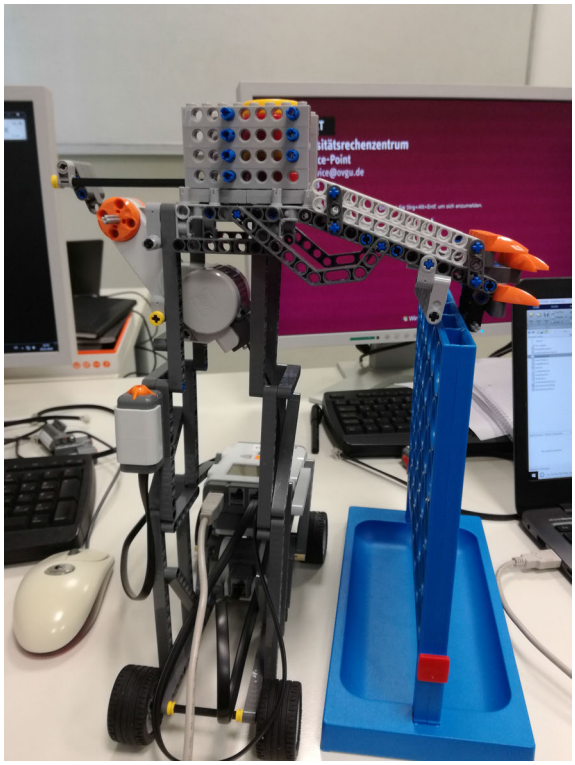

Abbildung 3. 4-Gewinnt-Roboter

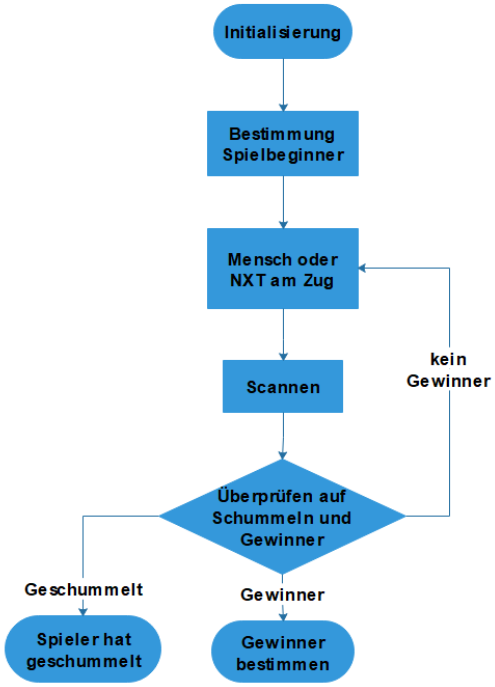

Abbildung 4. Programmablaufplan

Spielsteinlager befestigt. Dieses hatte nur Platz für 6 Spielsteine. Somit musste man während des Spieles immer neue Steine nachlegen. Über eine Rampe gelangten die Spielsteine dann in die einzelnen Zellen. Da die Rampe etwas höher lag als das Spielfeld, wurden am vorderen Teil der Rampe, 2 Leitschienen angebracht. So wurde sichergestellt, dass die Steine im Spielfeld landen. Eine weitere Notwendigkeit war es die Ränder des Spielfeldes etwas anzuschleifen um eine trichterartige Öffnung zu erzeugen. Dies hatte den Effekt, dass die Spielsteine nicht mehr am Rand stecken blieben. Zum Auswerfen der Steine wurde ein einfacher mechanischer Arm konstruiert. Dieser hatte eine festgelegte Schubdistanz. Nach dem Einwerfen fuhr der Roboter zweimal eine kurze Distanz vor und zurück. Dies sollte dabei helfen, Verkantungen von Spielsteinen zu verhindern. Außerdem wurde am Turm ein Tastsensor befestigt. Dieser wurde dazu verwendet dem Roboter zu signalisieren, dass er am Zug ist.

D. Funktionsweise des Programms

Sobald die Webcam ausgerichtet war, kann das Programm über die graphische Benutzeroberfläche initialisiert werden. Der komplette Programmablaufplan ist in Abbildung [4] dargestellt. Zunächst wurde zufällig der Spielbeginner festgelegt. Über diese graphische Benutzeroberfläche konnte auch die von der künstlichen Intelligenz erkannte Matrix überprüft werden. Dies ermöglichte eine Echtzeitüberprüfung. Began der menschliche Spieler, musste dieser zunächst seinen Spielzug ausführen. Wenn er mit diesem fertig war, drückte er den Tastsensor. Dies signalisierte dem Roboter, dass er am Zug ist. Daraufhin scannte dieser das Spielfeld und lies die in [III-B] erwähnte Prüfroutine durchlaufen. Wenn eins der Kriterien: Betrug, Rot gewinnt, Gelb gewinnt oder Unentschieden festgestellt wurde, wurde das Programm beendet. Eine entsprechende Anzeige über den Ausgang des Spieles wurde in der graphischen Benutzeroberfläche ausgegeben. Wenn keins dieser Kriterien wurde das Spiel fortgesetzt. Daraufhin berechnete der Algorithmus dann seinen Zug und führte diesen aus. Dieser Prozess wiederholte sich solange bis eins der spielbeendenden Kriterien erfasst wurde.

IV. Ergebnisdiskussion

Nach Vollendung des Projektseminars Lego Mindstorms war ein Roboter entwickelt worden, der autonom gegen einen menschlichen Gegenspieler 4-Gewinnt spielen konnte. In mehreren Testdurchläufen schaffte der Roboter es gegen den menschlichen Spieler zu gewinnnen. Jedoch versuchte der Mensch auch nicht die Gewinnsituationen des Roboters zu verhindern. Dies zeigte, dass der Roboter sich selber erfolgreich Gewinnsituationen kreieren konnte und auch gegnerische Gewinnsituationen verhinderte. Ein größeres Problem beim Spielen stellte hierbei jedoch die Erkennung durch die Webcam dar. Es musste aufgepasst werden, weder das Spielfeld noch die Webcam zu verschieben, weil sonst das Spiel nicht mehr erkannt wurde. Trat dies ein war eine Rekalibrierung der Webcam erforderlich. Diese war relativ zeitaufwendig und verzögerte das Spiel. Auch musste man bei der Webcam auf die Beleuchtung achten. Die Farberkennung war lichtabhängig, wodurch bei anderer Beleuchtung auch andere Farben erkannt wurden. Dies wurde teilweise umgangen indem die neuen Farbspektren in der regulären Farberkennung ergänzt wurden. Auch stellte das Spielsteinlager und das Einwerfen der Spielsteine ein Problem dar. Beim Einwerfen gab es anfangs das Problem, dass die

Rampe etwas höher gelegen war als das Spielfeld. Dadurch entstand eine Lücke, durch die die Spielsteine heraus fallen konnten. Dies wurde durch die in Abschnitt [III-C] erwähnten Leitschienen behoben. Beim Spielsteinlager wurde zuerst ein geschlossenes Lager gebaut. Bei diesem erwies es sich jedoch als schwierig die Steine hineinzufüllen. Ein weiteres Problem stellte das Verkanten der Steine dar. Deshalb wurde dann ein halboffenes Magazin verwendet. Dies hatte den Vorteil, dass es zum einen besser befüllbar war und zum anderen konnte man sehen, wenn sich Steine verkanteten. Somit konnte man das Problem schnell beheben. Eine weitere Hürde war die Ausrichtung des Roboters. Bei dieser musste darauf geachtet werden, dass der Roboter genau parallel zum Spielfeld fuhr, weil sonst im Verlaufe des Spieles eine zu hohe Abweichung entstand. Als Folge trafen die Spielsteine dann nicht ihre Zellen. Im schlimmsten Fall wurde sogar das Spielfeld verschoben.

V. Zusammenfassung und Fazit

Während des Projektseminars Lego Mindstorms wurde erfolgreich ein Roboter entwickelt, der autonom gegen einen menschlichen Gegenspieler 4-Gewinnt spielen konnte. Jedoch gibt es noch einige Verbesserungsmöglichkeiten. Eine Idee war es ein Area of Interest für die Webcam festzulegen. Dies würde die Kalibrierung erleichtern und den Zeitaufwand dafür deutlich verringern. Eine weitere Verbesserungsidee war es, den Roboter in einer festen Schiene fahren zu lassen. Dies würde den Vorteil bieten, dass man diesen nicht vor jedem Spiel neu ausrichten müsste. Auch könnten so Fehler und Ungenauigkeiten besser vermieden werden, da so Fehler vom Menschen ausgeschlossen wären. Das in [IV] beschriebene Spielsteinlager könnte auch noch ausgebessert werden, denn hier traten immernoch Verkantungen der Spielsteine auf. Hierzu wäre ein Lager, welches genauso breit ist wie die Spielsteine selbst, angebracht. Dies mit Lego zu konstruieren erwies sich jedoch als recht anspruchsvoll. Auch könnte man die Kapazität des Lagers noch auf 21 Spielsteine erhöhen, sodass man während des Spielens keine Steine mehr nachlegen müsste. So könnte man dann ein ganzes Spiel am Stück durchspielen. Eine andere Verbesserung wäre ein besserer Spielalgorithmus. Dieser gestaltete sich bei diesem 4-Gewinnt-Roboter relativ simpel. Mit einer richtigen Implementierung des MiniMax-Algorithmus könnte der Roboter ein besseres Spielverhalten aufweisen. Somit wäre es auch möglich die Suchtiefe des Algorithmus variabel einzustellen. Dies wäre eine Variante der Implementierung von verschiedenen Schwierigkeitsgraden. über die graphische Benutzeroberfläche könnte man eine solche Auswahlmöglichkeit realisieren.

Literatur- und Bildverzeichnis

[1] Wikipedia, Nullsummenspiel, https://de.wikipedia.org/wiki/Nullsummenspiel (Abruf: 15.03.2019)
[2] Wikipedia, Minimax-Algorithmus, https://de.wikipedia.org/wiki/Minimax-Algorithmus (Abruf: 15.03.2019)
[3] Christian Kanzow und Alexandra Schwartz *Spieltheorie: Theorie und Verfahren zur Lösung von Nash- und verallgemeinerten Nash-Gleichgewichtsproblemen, S.5-7*. 02.Oktober 2018.
[4] Hrsg. v. Görz, Günther / Schneeberger, Josef / Schmid, Ute *Handbuch der Künstlichen Intelligenz, S.624-627*. Oktober 2013.