Tic-Tac-Toe-Roboter

Marko Gaube, Elektrotechnik und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—In dieser Arbeit wird die Entwicklung und Realisierung eines LEGO-Roboters vorgestellt, der in der Lage ist, das Spiel Tic-Tac-Toe gegen einen menschlichen Gegner auf einem Blatt Papier zu spielen. Das Spielfeld wird mittels einer Kamera erfasst und durch einen Bildverarbeitungsalgorithmus ausgewertet, wobei die Erkennung der Spielerzüge auf einer Farbanalyse basiert. Zur Bestimmung des optimalen Spielzugs des Roboters kommt ein rekursiver Minimax-Algorithmus zum Einsatz. Die physische Umsetzung der Spielzüge erfolgt durch einen dreiachsigen Plotter auf Basis eines kartesischen Koordinatensystems. Grundlage bildet das LEGO-Mindstorms-System. Die Programmierung erfolgt in MATLAB.**

***Schlagwörter*—Bilderkennung, Minimax-Algorithmus, Plotter, Roboter, Tic-Tac-Toe**

I. Einleitung

Ziel dieses Projekts ist die Entwicklung eines Roboters, der in der Lage ist, das Spiel Tic-Tac-Toe gegen einen menschlichen Gegner zu spielen. Das Spiel wird dabei konventionell auf einem Blatt Papier durchgeführt. Daraus ergibt sich die Anforderung, dass der Roboter einen menschlichen Spielzug visuell erfassen, diesen verarbeiten und auf Basis des aktuellen Spielzustands einen eigenen Zug generieren muss. Weiterhin soll der Roboter in der Lage sein, das Spielfeld vor dem eigentlichen Spielbeginn selbstständig zu zeichnen sowie den Spielverlauf zu analysieren, um das Spielende und einen möglichen Gewinner korrekt zu erkennen.

Zur Umsetzung dieser Anforderungen wird das Spielfeld mithilfe einer Kamera erfasst. Der aktuelle Zustand des Spielfelds wird anschließend durch einen in MATLAB implementierten Algorithmus ausgewertet, welcher den optimalen nächsten Zug berechnet. Der physische Eintrag des berechneten Zuges erfolgt mittels eines Plotters auf dem Papier. Der fertige Roboter ist in Abbildung 1 dargestellt. Im Folgenden wird die konkrete Umsetzung erläutert.

II. Vorbetrachtungen

Das Spiel Tic-Tac-Toe ist allgemein bekannt und zeichnet sich durch einfache Regeln sowie eine überschaubare Spielstruktur aus. Trotz dieser Einfachheit ergeben sich bei der technischen Umsetzung als Tic-Tac-Toe-Roboter vielfältige Herausforderungen.

Im Rahmen des Projektseminars Elektrotechnik/Informationstechnik an der Otto-von-Guericke-Universität Magdeburg (OVGU) wurden bereits zahlreiche Realisierungskonzepte für Tic-Tac-Toe-Roboter sowie für Schreibroboter entwickelt. Ziel dieses Projekts ist es, beide Ansätze in einem integrierten System zu vereinen.

Der softwareseitige Schwerpunkt liegt auf der Implementierung des vollständigen Spielablaufs, einschließlich einer optimalen Zugstrategie für den Roboter wie in [1]. Die zentrale mechanische Herausforderung besteht hingegen in der Konstruktion eines zuverlässigen Plotters. Darüber hinaus soll durch die Einbindung einer Kamera ein erster Einblick in die Bildverarbeitung mit MATLAB ermöglicht werden.

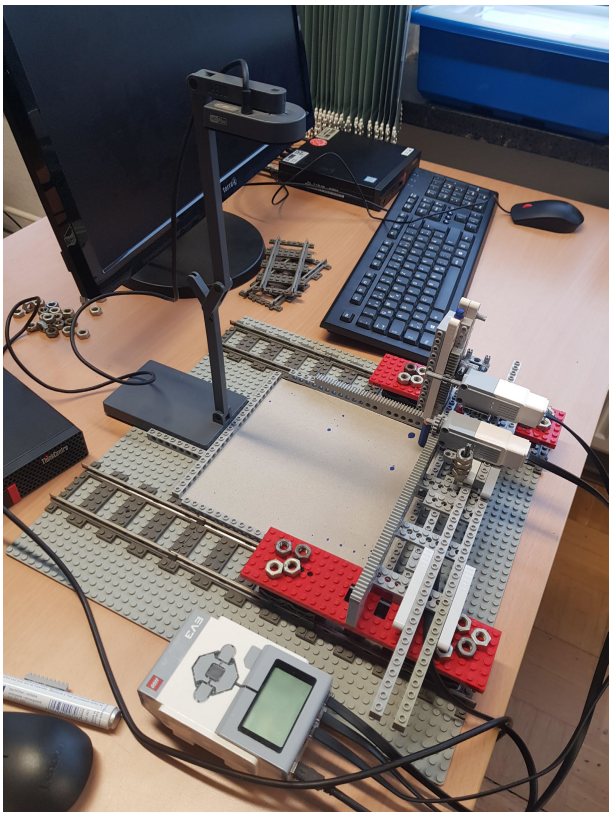

Abbildung 1. Fertiger Tic-Tac-Toe-Roboter

III. Umsetzung

A. Konstruktion des Plotters

Wie Abbildung 2 zeigt, basiert der Plotter auf einem dreidimensionalen kartesischen Koordinatensystem, wobei das Spielfeld in der xy-Ebene liegt. Dieses befindet sich direkt auf einer LEGO-Basisplatte, die die Grundlage der gesamten Konstruktion bildet. Zur Positionierung des Stiftes ist neben der Navigation in x- und y-Richtung zusätzlich eine Bewegung entlang der z-Achse erforderlich, um den Stift anzuheben und abzusetzen.

Für jede Raumrichtung wird ein separater Motor eingesetzt. Durch die Verwendung der kleinen LEGO-Motoren kann der Plotter kompakter gebaut und das Gewicht reduziert werden. Die Kraftübertragung erfolgt jeweils über Zahnräder, die

DOI: 10.24352/UB.OVGU-2026-042

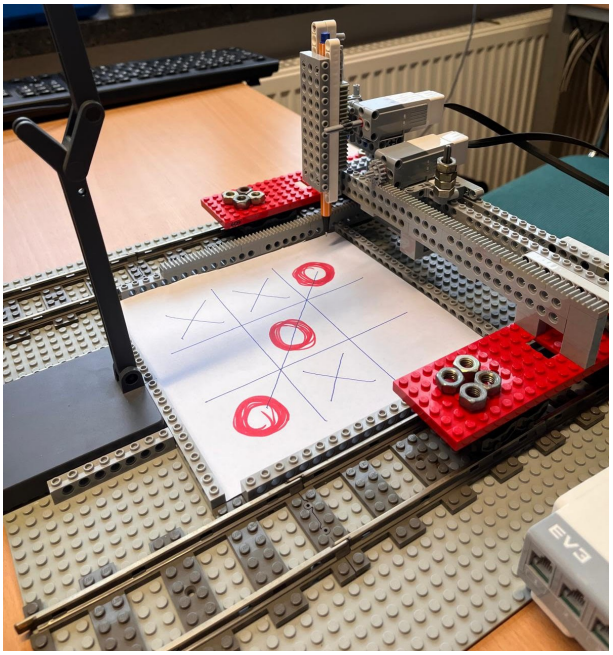

Abbildung 2. Der Plotter basiert auf einem kartesischen Koordinatensystem. Die Spitze des Stiftes befindet sich im Koordinatenursprung $(0, 0, 0)$.

in Zahnstangen eingreifen und so eine lineare Bewegung erzeugen. Dabei bewegt `motorX` einen Schlitten entlang der an beiden Seiten des Spielfeldes montierten LEGO-Schienen, um eine Auslenkung in x-Richtung zu ermöglichen. Auf diesem Schlitten ist eine weitere Führungsschiene aus LEGO-Steinen angebracht, auf der der `motorY` läuft. Der `motorZ` ist wiederum direkt am `motorY` befestigt und ermöglicht das Aufsetzen und Anheben des Filzstifts mittels einer Führung parallel zur z-Achse.

Die Positionssteuerung erfolgt durch die Übergabe von Drehwinkeln an die jeweiligen Motoren. Aufgrund der mechanischen Übersetzung zwischen Zahnrad und Zahnstange ergibt sich daraus eine lineare Verschiebung entlang der entsprechenden Achse, die proportional zum Drehwinkel ist. Die maximal zulässige Auslenkung in jede Richtung wird softwareseitig durch in Variablen gespeicherte Grenzwerte festgelegt. An die implementierte Funktion `movePen()` werden Bruchteile dieser Grenzwerte übergeben, welche in einer Auslenkung um eine entsprechende Teilstrecke innerhalb des definierten Spielfeldes resultieren.

B. Spielablauf

Der Programmablauf in der Hauptdatei `main` wird in Abbildung 3 verdeutlicht. Die `main`-Datei fasst den gesamten Spielprozess zusammen und koordiniert sämtliche Funktionsaufrufe.

Zu Beginn erfolgt die Initialisierung des Systems. In diesem Schritt werden der LEGO-EV3-Stein mit den drei Motoren sowie die Kamera in das Programm eingebunden. Zusätzlich wird eine 3×3-Matrix `playingField` als zentrale Datenstruktur initialisiert, die den aktuellen Zustand des Spielfeldes repräsentiert. Die Matrixeinträge sind wie folgt codiert: Der Wert 0 kennzeichnet ein freies Feld, 1 ein vom Roboter gesetztes Kreuz und 2 einen vom menschlichen Spieler gesetzten Kreis. Das Array `freeFields` speichert alle freien Felder. Außerdem wird die Größe des Spielfelds definiert und das Raster durch den Plotter auf das Papier aufgebracht.

Der eigentliche Spielablauf ist als Endlosschleife implementiert, das heißt, die Bedingung im Kopf der while-Schleife wurde auf `true` gesetzt. Diese wird nur beim Eintreten einer Abbruchbedingung innerhalb der Schleife verlassen. Das Spiel beginnt mit dem Zug des Roboters. Hierzu wird mit der Matrix `playingField` der aktuelle Spielfeldzustand an den Minimax-Algorithmus übergeben, der den optimalen Zug bestimmt. Die berechnete Position wird in der Matrix mit dem Wert 1 aktualisiert und physisch als Kreuz auf das Spielfeld geplottet. Anschließend erfolgt eine Überprüfung auf die Abbruchbedingungen. Diese gelten als erfüllt, wenn

1) eine Gewinnkonstellation für den Roboter vorliegt (wenn der Wert 1 dreimal in einer Zeile, Spalte oder Diagonale der Matrix auftritt), oder
2) keine freien Felder mehr verfügbar sind (`freeFields` keine Elemente enthält).

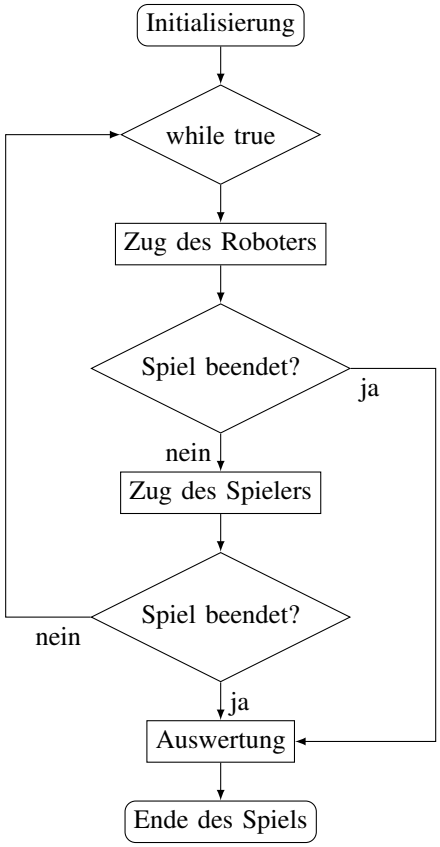

Abbildung 3. Programmablaufplan für den Spielablauf in der `main`-Datei

Ist das Spiel noch nicht beendet, folgt der Zug des menschlichen Spielers. Dieser markiert sein Feld durch das Zeichnen eines roten Kreises auf dem Papier. Die Position des Kreises wird mittels Kamerabild und eines Algorithmus zur Bilderkennung erfasst. Der erkannte Zug wird in der Matrix mit dem Wert 2 gespeichert. Anschließend werden analog zum vorherigen Schritt die Abbruchbedingungen geprüft. Allerdings

wird die Matrix `playingField` auf den Wert 2 hin überprüft, um einen eventuellen Gewinn des Spielers festzustellen.

Beim Eintreten einer Abbruchbedingung wird die Schleife durch eine `break`-Anweisung verlassen und eine abschließende Auswertung durchgeführt. Im Falle eines Gewinns („drei in einer Reihe“) markiert der Plotter diesen durch das Ziehen eines Strichs und gibt den Gewinner im MATLAB-Befehlsfenster aus; andernfalls wird das Spiel als Unentschieden beendet.

C. Bilderkennung

Zu Beginn des Spielerzugs wird der Programmablauf für 15 Sekunden pausiert, um dem menschlichen Spieler die Möglichkeit zu geben, seinen Zug durch das Einzeichnen eines roten Kreises auf dem Spielfeld zu kennzeichnen. Anschließend wird mithilfe der oberhalb des Spielfelds angebrachten Kamera eine Aufnahme des gesamten Spielfelds erstellt. Die Kamera kann durch die „Image Acquisition Toolbox“ in das Programm integriert werden [2].

Weiterhin stellt MATLAB Funktionen zur rudimentären Bilduntersuchung bereit [3]. Jedem Pixel eines Bildes kann eine eindeutige x- und y-Koordinate zugeordnet werden, welche seine Position innerhalb der Bildebene beschreibt. Dadurch ist eine feldweise Bildauswertung möglich, indem Bildausschnitte an vordefinierten Positionen, die mit den einzelnen Spielfeldern übereinstimmen, ausgewertet werden. Die Positionsbestimmung anhand der x- und y-Koordinaten wird in Abbildung 4 verdeutlicht.

Für die Detektion eines Spielerzugs wird nicht die geometrische Form des Kreises analysiert, sondern ausschließlich dessen Farbwert. In jedem definierten Bildausschnitt wird die Anzahl roter Pixel ermittelt. Ein Feld wird als gültiger Spielerzug gewertet, wenn darin mehr als 200 rote Pixel gezählt werden. Ein Pixel wird dabei als „rot“ klassifiziert, wenn es einen RGB-Farbcode von $(R > 150,\ G < 100,\ B < 100)$ aufweist. Das entsprechend identifizierte Feld wird im Array `playingField` mit dem Wert 2 markiert.

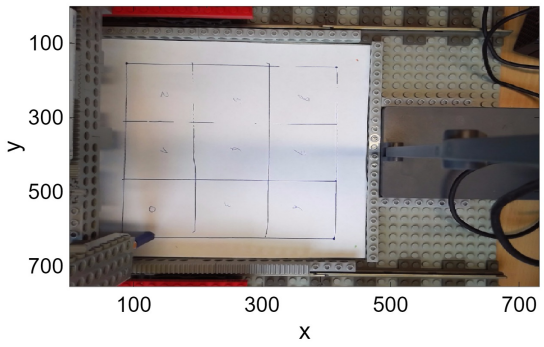

Abbildung 4. Kamerabild des Spielfelds mit geplottetem Raster. Die Positionsbestimmung der Bildausschnitte erfolgt mithilfe der Funktion `impixelinfo()`.

D. Minimax-Algorithmus

Tic-Tac-Toe zählt zu den diskreten Zwei-Spieler-Spielen. Diese Klasse von Spielen kann durch einen Minimax-Algorithmus optimal gelöst werden [4]. Die Implementierung des Algorithmus orientiert sich an dem in [5] diskutierten Programmierbeispiel und wird im Weiteren erläutert:

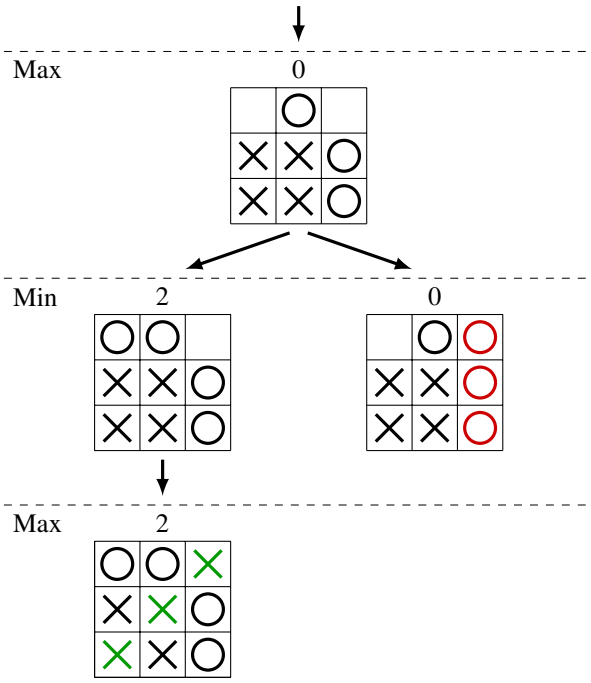

Abbildung 5. Ausschnitt aus einem Spielbaum mit Visualisierung des Minimax-Algorithmus, grün: Gewinn des Computers, rot: Gewinn des Gegenspielers

Der aktuelle Spielzustand wird dem Minimax-Algorithmus als Matrix `playingField` übergeben. Dieser Zustand wird bewertet. Dabei wird ein Gewinn des Roboters mit dem Wert 2, ein Unentschieden mit dem Wert 1 und ein Gewinn des menschlichen Spielers mit dem Wert 0 gleichgesetzt.

Liegt kein terminaler Spielzustand vor, erfolgt keine unmittelbare Bewertung. Stattdessen generiert der Algorithmus rekursiv alle möglichen Folgezustände, indem er sämtliche zulässigen Züge simuliert. Für jeden dieser Zustände wird der Minimax-Algorithmus erneut aufgerufen, sodass virtuell ein vollständiger Spielbaum erstellt wird.

Die Auswahl des optimalen Zugs erfolgt gemäß der Minimax-Strategie: Befindet sich der Roboter am Zug (Max-Knoten), wird der maximale Bewertungswert der Folgezustände gewählt. Ist hingegen der menschliche Spieler am Zug (Min-Knoten), wird der minimale Bewertungswert ausgewählt. Dieses Vorgehen basiert auf der Annahme, dass beide Spieler optimal spielen.

Wie in Abbildung 5 verdeutlicht, werden dadurch Spielzüge vermieden, die zwar schließlich zu einem Gewinn führen würden, jedoch dem Gegner einen früheren Gewinn ermöglichen.

IV. Ergebnisdiskussion

Zum Abschluss des Projekts war der Roboter in der Lage, mehrere Spielabläufe zuverlässig nacheinander auszuführen. Dessen grundsätzliche Funktionsfähigkeit konnte damit bestätigt werden.

Die softwareseitige Umsetzung des Plotters gestaltet sich vergleichsweise einfach. Da dieser auf einem kartesischen

Koordinatensystem basiert und ausschließlich Geraden zeichnet, ist die Ansteuerung der Motoren mit geringem mathematischen Aufwand umsetzbar. Demgegenüber weist die mechanische Konstruktion eine höhere Fehleranfälligkeit auf. Insbesondere auftretende Reibungskräfte an den Führungsschienen sowie zwischen Stift und Papier können die Positionsgenauigkeit beeinträchtigen. Durch die Verwendung von LEGO-Schienen ließ sich diese Problematik reduzieren. Zusätzlich führte das Anbringen von Gewichten zu einer erhöhten Bahnstabilität und damit zu einer verbesserten Präzision des Plotters.

MATLAB stellt mit der „Image Acquisition Toolbox“ umfangreiche Funktionen zur Verfügung, die die Bildverarbeitung ermöglichen. Bei der Implementierung der Zugerkennung zeigte sich jedoch, dass ein geeigneter RGB-Schwellenwert und Grenzwert für die Anzahl roter Pixel stark von der Wahl der Kamera, der Stiftfarbe und der Beleuchtungssituation abhängen. Daher mussten diese Parameter empirisch bestimmt werden, um eine zuverlässige Spielerzugerkennung zu gewährleisten.

V. Zusammenfassung und Fazit

Das zu Beginn formulierte Ziel, einen Roboter zu bauen, der ein komplettes Tic-Tac-Toe-Spiel auf Papier gegen einen Menschen spielt, wurde erreicht. Dazu wurden hauptsächlich folgende Komponenten verwendet:

1) LEGO-EV3-Stein
2) drei kleine LEGO-Motoren
3) Dokumentenkamera
4) MATLAB-Software
5) LEGO-Steine
6) LEGO-Schienen

Da der Roboter durch den Minimax-Algorithmus optimal spielt, kann dieser nicht besiegt werden. Allenfalls ist ein Unentschieden bei einer ebenfalls optimalen Strategie des menschlichen Gegenspielers erreichbar.

Darüber hinaus wäre z. B. zusätzlich eine Auswahl des beginnenden Spielers und eine Einstellung für den Schwierigkeitsgrad des Roboters denkbar. Außerdem wäre eine Erweiterung auf ein „Vier Gewinnt“-Spiel realisierbar, da sowohl die Ansteuerung des Plotters als auch der Minimax-Algorithmus entsprechend leicht adaptiert werden könnten.

Literaturverzeichnis

[1] Schulz, Karsten: *Der Tic-Tac-Toe-Robo*. https://journals.ub.ovgu.de/index.php/LEGO/article/view/2262/2242. Version: Februar 2026

[2] The MathWorks, Inc.: *Image Acquisition Toolbox — Functions*. https://de.mathworks.com/help/releases/R2025b/imaq/referencelist.html?type=function&s_tid=CRUX_topnav. Version: Februar 2026

[3] The MathWorks, Inc.: *Build Interactive Tools — Functions*. https://de.mathworks.com/help/images/referencelist.html?type=function&s_tid=CRUX_topnav&category=building-guis-with-modular-interactive-tools. Version: Februar 2026

[4] Nopper, Tobias: *Prinzip MinMax-Algorithmus am Beispiel von Tic Tac Toe | Wie spielen Computer?* https://youtu.be/3ufcmCpKb6w. Version: Februar 2026

[5] Nopper, Tobias: *Implementierung des MinMax-Algorithmus für Tic Tac Toe | Wie spielen Computer?* https://youtu.be/ODgPsXssUvk. Version: Februar 2026