Automatisierung des Spiels Tic-Tac-Toe

Finn Löffler, Elektromobilität
Otto-von-Guericke-Universität Magdeburg

***Zusammenfassung*—Dieses Paper beschreibt die Entwicklung eines Tic-Tac-Toe-Roboters während eines zweiwöchigen Erstsemester-Projektseminars. Das Ziel war ein System, das selbstständig gegen einen Menschen spielen kann. Das Paper zeigt den Verlauf von einem einfachen Modell mit Tastaturbedienung und Zufallszügen hin zu einem System mit automatischer Farberkennung mittels Kamera und der Implementierung eines Minimax-Algorithmus zur Bestimmung des optimalen Spielzugs des Roboters. Im vorgegebenen Zeitrahmen konnte ein funktionsfähiger Roboter realisiert werden, der Spielzüge erkennt und eigenständig auf Papier ausführt.**

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

I. Einleitung

Im Rahmen des zweiwöchigen Projektseminars wurde die Aufgabe gestellt, einen Roboter zu entwickeln. Die Wahl fiel auf einen Tic-Tac-Toe-Roboter, da dieses Projekt eine spannende Kombination aus Mechanik und Programmierung darstellt. Das Ziel bestand darin, einen Roboter zu konstruieren, der gegen einen menschlichen Gegner antreten kann. Das fertige System ist in Abbildung 1 dargestellt. In diesem Bericht wird beschrieben, wie der Roboter geplant, umgesetzt und schrittweise verbessert wurde, um eine funktionierende Spielumgebung zu realisieren.

II. Vorbetrachtungen

Als Hardware-Plattform wurde das LEGO-Mindstorms-Bausystem vorgegeben. Die mechanische Umsetzung erfolgte in Form eines Plotters. Da das Thema Tic-Tac-Toe bereits in früheren Seminaren behandelt wurde (siehe [1]), lag die Herausforderung in der Integration von Bilderkennung und Spielalgorithmus in ein stabiles mechanisches System. Im Prozess wurde die Steuerung von einer manuellen Tastatureingabe auf eine automatisierte Erkennung mittels Dokumentenkamera umgestellt und die Spiel-Logik sukzessive von Zufallszügen auf eine strategische Spielweise optimiert.

III. Umsetzung

A. Konstruktion

Bei der Konstruktion des Tic-Tac-Toe-Roboters wurde sich für ein Plotter-Design entschieden. Dies hat wesentliche Vorteile: Einerseits wurde hierdurch eine gleichbleibend hohe Qualität der Spielabläufe gewährleistet, da sich ein identischer Startpunkt für jeden Spieldurchlauf festlegen ließ. Andererseits konnte durch das gewählte Design die Reibung auf ein Minimum reduziert werden.
Der Plotter verfügte über drei Achsen, die jeweils von einem kleinen Motor angetrieben wurden, wie in Abbildung 2 zu sehen ist. Die Kraftübertragung erfolgte jeweils über Zahnräder, die in Zahnstangen eingriffen und so eine lineare Bewegung erzeugten.

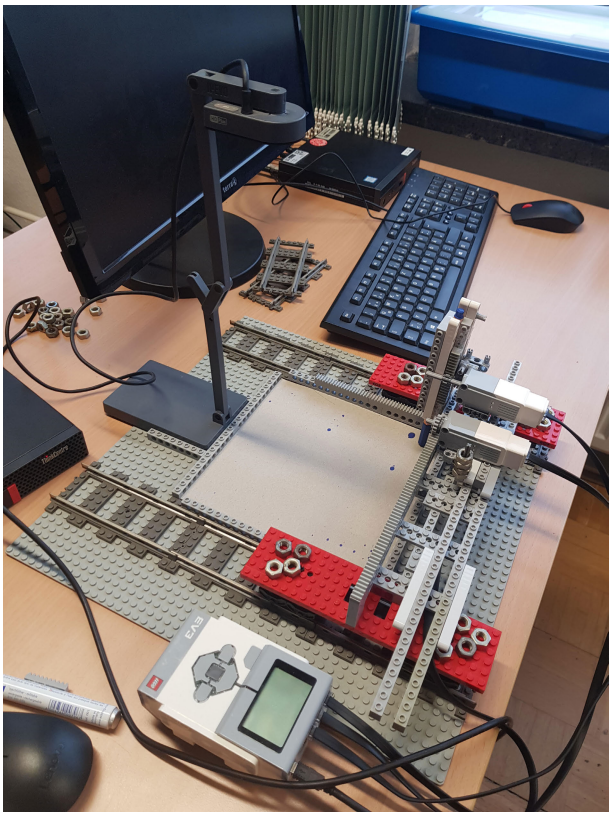

Abbildung 1. Fertiger Tic-Tac-Toe-Roboter

Der Roboter navigierte auf der x- und y-Achse über das Spielfeld, während der Stift auf der z-Achse angehoben und gesenkt wurde. Mit dem Stift wurde das Spielfeld sowie die Spielzüge des Computers aufgezeichnet. Aufgrund der niedrigen Reibung des gewählten Designs konnten kleine LEGO-Mindstorms-Motoren verwendet werden. Diese ließen sich aufgrund ihrer Größe und Form besonders leicht in das Design integrieren. Zur weiteren Reduzierung der Reibung wurden Schienen für die Bewegung auf der x-Achse verbaut und auf einer quadratischen LEGO-Grundplatte befestigt, was die Stabilität des Roboters zusätzlich erhöhte.
Der Stift wurde in einer Vorrichtung fixiert, welche die vertikale Bewegung präzise ausführte. Im weiteren Projektverlauf wurde eine Kamera ergänzt; um den Bildausschnitt konstant zu halten, wurde auch diese mit flachen LEGO-Steinen auf der Grundplatte fixiert. Da eine maßgeschneiderte Lösung zeitlich

DOI: 10.24352/UB.OVGU-2026-043

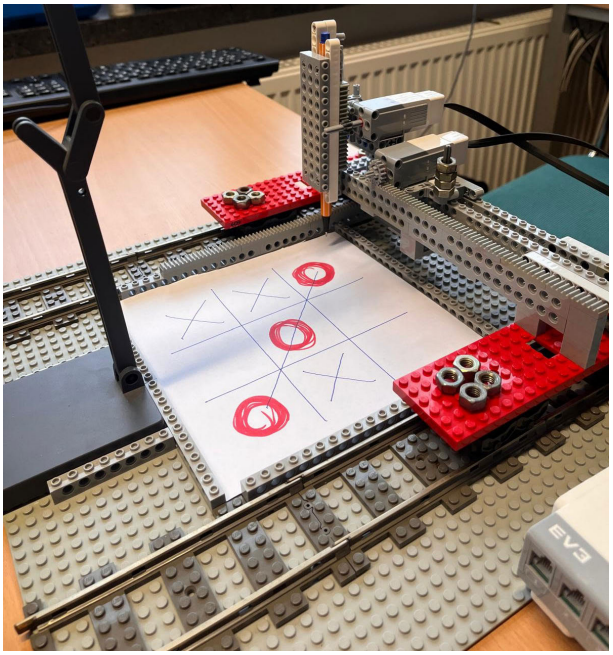

Abbildung 2. Konstruktion des Plotters

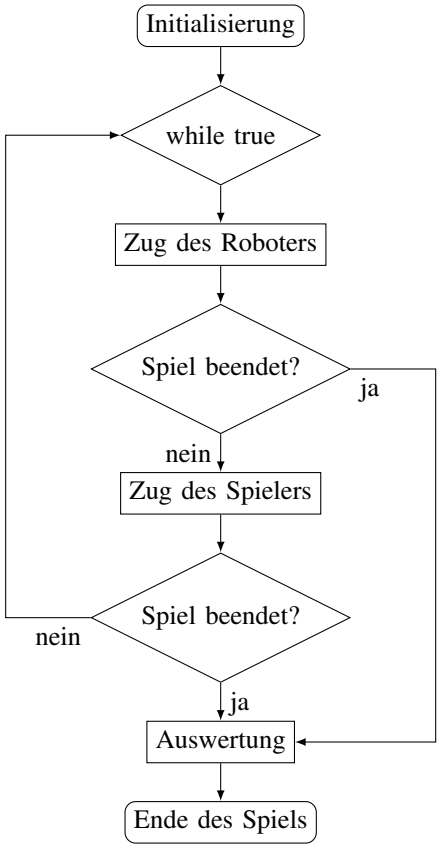

Abbildung 3. Programmablaufplan für den Spielverlauf im Hauptprogramm

nicht mehr umsetzbar war, wurden Muttern als unfixierte Gewichte verwendet, um die Qualität der Linienzüge zu gewährleisten. Hätte mehr Zeit zur Verfügung gestanden, hätten die Muttern durch eine technisch ausgereiftere Lösung ersetzt werden können.

B. Spielverlauf

Der Aufbau der Software, deren Logik durch den Programmablaufplan in Abbildung 3 verdeutlicht wird, basierte auf einem modularen Konzept. Dabei wurden sämtliche Abläufe, wie etwa die Erstellung des Spielfelds, in eigenständige Funktionen ausgelagert und erst innerhalb des Hauptprogramms miteinander verknüpft. Zu Programmbeginn wurde die EV3-Steuereinheit initialisiert, um potenziellen Fehlfunktionen der Winkelbegrenzung vorzubeugen. Zur Steuerung der Motoren wurden zudem sämtliche Parameter für Winkel und Geschwindigkeiten zentral in der Hauptfunktion als Variablen definiert. Diese Methode steigerte die Effizienz bei Anpassungen und minimierte das Fehlerrisiko, da Änderungen nicht in jeder Funktion einzeln vorgenommen werden mussten. Abschließend wurde eine 3×3-Matrix generiert, die als digitales Abbild des Spielfelds diente und den Wert 0 als Platzhalter für unbesetzte Felder nutzte.
Im darauffolgenden Schritt erfolgte die grafische Ausgabe des Spielfelds. Der Computer initiierte das Spiel, indem er seinen Zug vollzog und ein Kreuz in eines der neun unbesetzten Felder zeichnete. Dieser Vorgang wurde digital in der Matrix durch den Wert 1 sowie eine entsprechende Ausgabe in der Kommandozeile gekennzeichnet.
Im Anschluss setzte der Spieler seinen Zug, indem er einen roten Kreis in eines der verbliebenen freien Felder zeichnete, was systemseitig als 2 registriert wurde. Sobald der Spieler seinen Zug beendet hatte, identifizierte der Roboter das entsprechende Feld und reagierte mithilfe des Minimax-Algorithmus optimal auf die Eingabe. Die Spielroutine wurde so lange fortgesetzt, bis entweder eine der beiden Parteien den Sieg errang oder durch die Belegung sämtlicher Felder ein Unentschieden eintrat. Im Falle eines Sieges zeichnete der Roboter eine Linie durch die drei gewinnbringenden Felder und gab den Gewinner in der Kommandozeile aus. Standen keine freien Felder mehr zur Verfügung, deklarierte das System die Partie als Unentschieden.

C. Bilderkennung

Durch den Einsatz einer Dokumentenkamera wurde die eigenständige Erkennung sowie die Reaktion auf den Spielzug des menschlichen Spielers ermöglicht. Die Kamera konnte mithilfe der „Image Acquisition Toolbox“ in das Programm integriert werden [2]. Darüber hinaus stellt MATLAB Funktionen zur Bildanalyse bereit [3].
Der Bildausschnitt wurde zunächst digital unter Verwendung der Bildkoordinaten in neun gleich große Felder unterteilt. Diese Felder stellten in ihrer Gesamtheit das Spielfeld dar. Es wurde dabei auf die gleiche Ausrichtung der Kamera über mehrere Partien hinweg geachtet, damit die über die Koordinaten festgelegte Lage der Felder stets mit der realen Lage der Felder auf dem Spielfeld übereinstimmte. Der durch die Dokumentenkamera erfasste Ausschnitt ist in Abbildung 4 dargestellt.
Nachdem der Roboter seinen Zug ausgeführt hatte, wurde dem Spieler ein Zeitraum von 15 Sekunden eingeräumt. Innerhalb dieses Zeitraums markierte der Spieler seinen Zug in Form

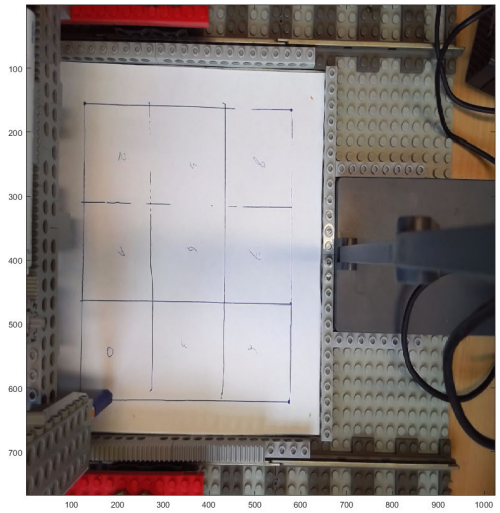

Abbildung 4. Kamerabild des Spielfelds mit aufgetragenem Spielfeldraster zur Positionsbestimmung der einzelnen Felder

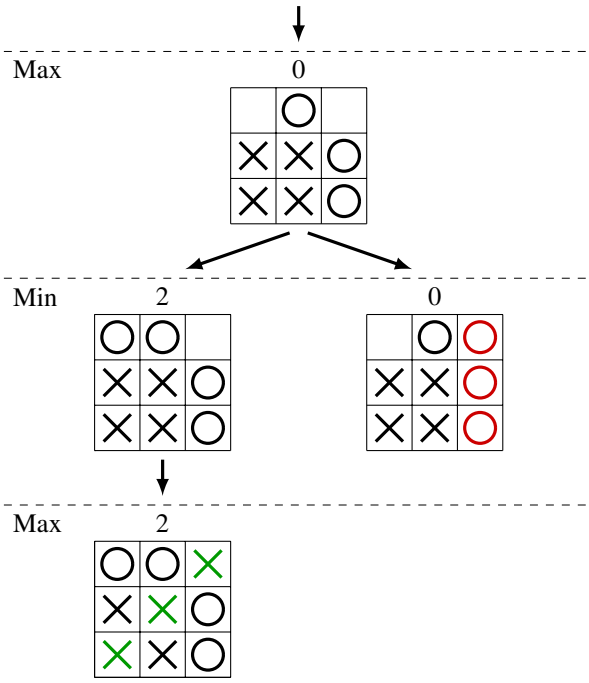

Abbildung 5. Ausschnitt aus einem Spielbaum mit Visualisierung des Minimax-Algorithmus, grün: Gewinn des Computers, rot: Gewinn des Gegenspielers

eines roten Kreises in einem der freien Felder. Nach Ablauf der vorgegebenen Zeit wurde ein Bild des Spielfelds durch die Dokumentenkamera aufgenommen und anschließend vom Computer ausgewertet. Dabei wurde nicht die Form des Kreises, sondern dessen Farbe analysiert. Ein Pixel wurde dabei als rot gewertet, sofern er den RGB-Farbcode $> 150, < 100, < 100$ aufwies. Die Voraussetzung, dass ein Zug einem Feld eindeutig zugeordnet werden konnte, beruhte auf der Messung der Anzahl von roten Pixeln innerhalb eines Feldbereichs. Wurde der Schwellenwert von 200 Pixeln in einem Feld überschritten, wurde das Feld als gültiger Spielzug gewertet.

D. Minimax- Algorithmus

Tic-Tac-Toe zählt zu den diskreten Zwei-Spieler-Spielen. Diese Klasse von Spielen lässt sich optimal durch einen Minimax-Algorithmus lösen [4].
Die Implementierung des Algorithmus orientierte sich an dem im Youtube-Video [5] diskutierten Programmierbeispiel. Dies wird im Folgenden erläutert:
Zur Bestimmung des idealen Spielzugs wurde dem Verfahren der aktuelle Status des Spielfelds in Form einer 3×3-Matrix übermittelt, woraufhin eine systematische Bewertung dieses Zustands erfolgte. Da die Komplexität des Spielbaums mit ca. $9!$ (362 880) möglichen Pfaden vergleichsweise gering ausfällt, konnte eine vollständige Exploration des Zustandsraums in Echtzeit realisiert werden.
Ein Sieg des Roboters wurde dabei mit dem Wert 2, ein Unentschieden mit 1 und ein Erfolg des menschlichen Kontrahenten mit 0 definiert.
Sofern noch kein Endzustand vorlag, wurde zunächst keine unmittelbare Gewichtung vorgenommen. Stattdessen generierte der Algorithmus rekursiv sämtliche denkbaren Folgeszenarien bis zum jeweiligen Spielende, indem alle zulässigen Züge simuliert wurden. Für jeden dieser Zweige wurde der Ablauf erneut angestoßen, um den gesamten Entscheidungsbaum zu erfassen.
Die Festlegung auf den günstigsten Pfad folgte einer dualen Strategie: War der Roboter am Zug, wurde stets der höchstmögliche Ergebniswert aus den verfügbaren Optionen favorisiert, während bei den simulierten Zügen des Menschen das geringste Resultat berücksichtigt wurde. Dieses Vorgehen basierte auf der Prämisse, dass beide Parteien fehlerfrei agieren. Der Algorithmus fungierte hierbei als Werkzeug zur Lösung des Spiels, indem er bewies, dass bei beidseitig perfekter Strategie jedes Spiel zwangsläufig in einem Unentschieden (Wert 1) endet.
Dadurch ließen sich Pfade ausschließen, die zwar theoretisch zum Erfolg führen könnten, jedoch ein suboptimales Spielverhalten des Gegners voraussetzen würden.
Abbildung 5 illustriert die Funktionsweise des Minimax-Algorithmus anhand eines Spielbaums. Ausgehend vom Status Quo an der Wurzel (oben) war der menschliche Spieler am Zug. Der Algorithmus berechnete vorausschauend, dass der Gegner bei optimaler Spielweise den rechten Pfad wählen würde, um sofort zu gewinnen (rote Markierung, Wert 0). Ein Sieg des Computers war nur möglich, falls der Mensch suboptimal agierte und den linken Pfad wählte; in diesem Fall gewann der Roboter im darauffolgenden Zug (grüne Markierung auf der 3. Ebene, Wert 2). Da der Minimax-Algorithmus von einem rationalen Gegner ausging, wurde der minimale Wert (0) an den Wurzelknoten (oben) zurückgegeben.

IV. Ergebnisdiskussion

Bis zum Ende des 14-tägigen Seminars konnte der Roboter mehrere Spielabläufe zuverlässig nacheinander absolvieren. Die Bilderkennung erwies sich jedoch als ausbaufähig, da Züge des Spielers am Rand eines Feldes gelegentlich nicht zuverlässig erkannt wurden. Der Minimax-Algorithmus funktionierte hingegen einwandfrei. Ein wiederkehrendes Problem stellten die auftretenden Reibungskräfte dar: Die Reibung zwischen Stift und Papier führte vereinzelt dazu, dass gezeichnete Linien einen Knick aufwiesen oder unvollständig blieben.

V. Zusammenfassung und Fazit

Im Rahmen des Projekts ist es gelungen, einen zuverlässig funktionierenden Tic-Tac-Toe-Roboter in Form eines Plotters zu konstruieren. Dabei wurden Reibung und Balance des Roboters sukzessive optimiert. Im Projektverlauf gelang der Übergang von einer manuellen Tastatureingabe und zufallsbasierten Spielzügen hin zu einer automatisierten Bilderkennung sowie einer strategischen Spielweise mittels Minimax-Algorithmus.

Trotz der Optimierung boten Reibung und Balance weiteren Verbesserungsbedarf, um zukünftig auf die Gewichte in Form von Muttern verzichten zu können. Auch bei der Software gibt es Möglichkeiten zur Weiterentwicklung. Zudem könnten die Spielstärke des Roboters sowie der Startspieler konfigurierbar gestaltet werden.

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