

LEGO-Farbsortiermaschine

Fabian Paschek, Elektro- und Informationstechnik
Otto-von-Guericke-Universität Magdeburg

Zusammenfassung—Im Rahmen des LEGO-Praktikums sollten die Studierenden nicht nur das theoretische Programmieren mit MATLAB, einer Software zur Lösung mathematischer Probleme und zur grafischen Darstellung der Ergebnisse, erlernen, sondern sich auch kreativ und praktisch mit dem Bau von selbstgewählten LEGO Robotern betätigen. Dabei entstand während des zweiwöchigen Praktikums eine Farbsortiermaschine, die mit Hilfe verschiedener LEGO-Mindstorms-Sensoren Objekte nach ihrer Farbe sortieren kann.

Schlagwörter—Automatisierung, Farbsortiermaschine, LEGO-Mindstorms, Lego-Praktikum, MATLAB, NXT

I. EINLEITUNG

MIT der Erfindung des Computers und dem immer schnelleren Fortschritt der Technik wurde die Entwicklung von Robotern nahezu mühelos und rasant vorangetrieben. Heute werden Roboter in vielen Bereichen der Industrie und des menschlichen Lebens eingesetzt. Die Aufgaben von Robotern sind dabei vielfältig, da sie Anforderungen mit hoher Präzision, Schnelligkeit und geringer Fehlerquote erfüllen können.

Doch was passiert, wenn in einem industriellen Fertigungsprozess fehlerhafte Produkte entstehen, die in Größe, Form oder Farbe voneinander abweichen?

Um eine gleichbleibend hohe Qualität eines Produktes zu gewährleisten, muss fehlerhafte Ware erkannt und aussortiert werden. Diese Qualitätsprüfung erfolgt meist noch durch Menschenhand, in zeitaufwendiger und mühsamer Arbeit. Dies kann jedoch schnell zu Messungenauigkeiten und Fehlern führen, weshalb auch in der Qualitätskontrolle zunehmender Roboter die menschliche Arbeitskraft ersetzen. Mit modernster Bildverarbeitung, Sensorik und künstlicher Intelligenz vermessen und überprüfen sie große Warenmengen und sortieren fehlerhafte Produkte in einem automatisierten Prozess aus. So stellen sie eine gleichbleibende Qualität der zu produzierenden Produkte sicher.

Aus der Faszination für Sortierroboter entstand das Ziel, eine Maschine zu entwickeln, die in der Lage ist, die Farben verschiedener Objekte zu erkennen und diese in dafür vorgesehene Kisten zu sortieren.

II. VORBETRACHTUNGEN

A. Was soll die Sortiermaschine alles können?

Die Sortiermaschine soll verschiedene Objekte nach ihrer Farbe in bestimmte, für die jeweilige Farbe vorgesehene Kisten sortieren. Dazu werden die Objekte auf ein Förderband gelegt, das durch einen Startknopf in Bewegung gesetzt wird. Eine Kamera soll dann ein Bild des Gegenstandes erstellen, aus dem eine Farbe extrahiert werden soll. Wird im Kamerabild eine Farbe erkannt, soll die jeweilige Box unter das Förderband

gefahren werden, so dass das Objekt mit der Farbe in die für diese Farbe vorgesehene Box fällt. Um Fehlsortierungen zu vermeiden, müssen also auch mehrfarbige Objekte erkannt und aussortiert werden.

B. Welche Komponenten werden dafür benötigt?

Zur Realisierung der Sortiermaschine steht ein LEGO-Mindstorms-Set mit verschiedenen Sensoren und Motoren zur Verfügung. Die zentrale Steuereinheit bildet der NXT-Stein, an den alle eingebauten Motoren und Sensoren angeschlossen sind. Die Programmierung des NXT-Steins erfolgt mit der Programmiersoftware MATLAB, die es mit Hilfe der Toolbox der RWTH Aachen ermöglicht, die Sensoren und Motoren über die MATLAB-Umgebung anzusprechen und zu konfigurieren. Zusätzlich dient ein Taster als Startknopf und eine Kamera nimmt Bilder der Objekte auf. Für die exakte Positionierung der Sortierboxen wird ein Ultraschallsensor verwendet. Damit sich nun die Kästen in die richtige Position bewegen und das Förderband die Objekte transportieren kann, sind zwei NXT-Motoren verbaut.

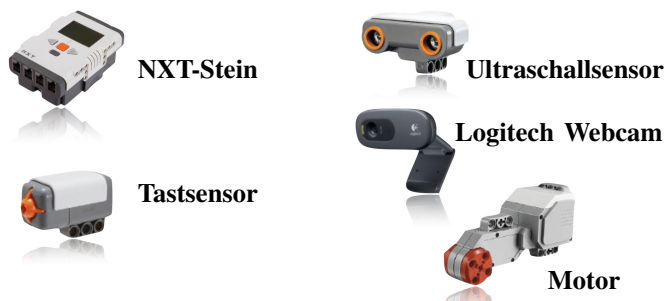


Abbildung 1. LEGO-Mindstorms-Komponenten entnommen aus [4]–[8]

III. UMSETZUNG

Im Folgenden soll die Umsetzung von der Konstruktion über die Funktionsweise bis hin zur Programmierung der Farbsortiermaschine erläutert werden.

A. Funktionsweise

Der folgende Programmablaufplan in Abbildung 2 zeigt die Funktionsweise der Sortiermaschine.

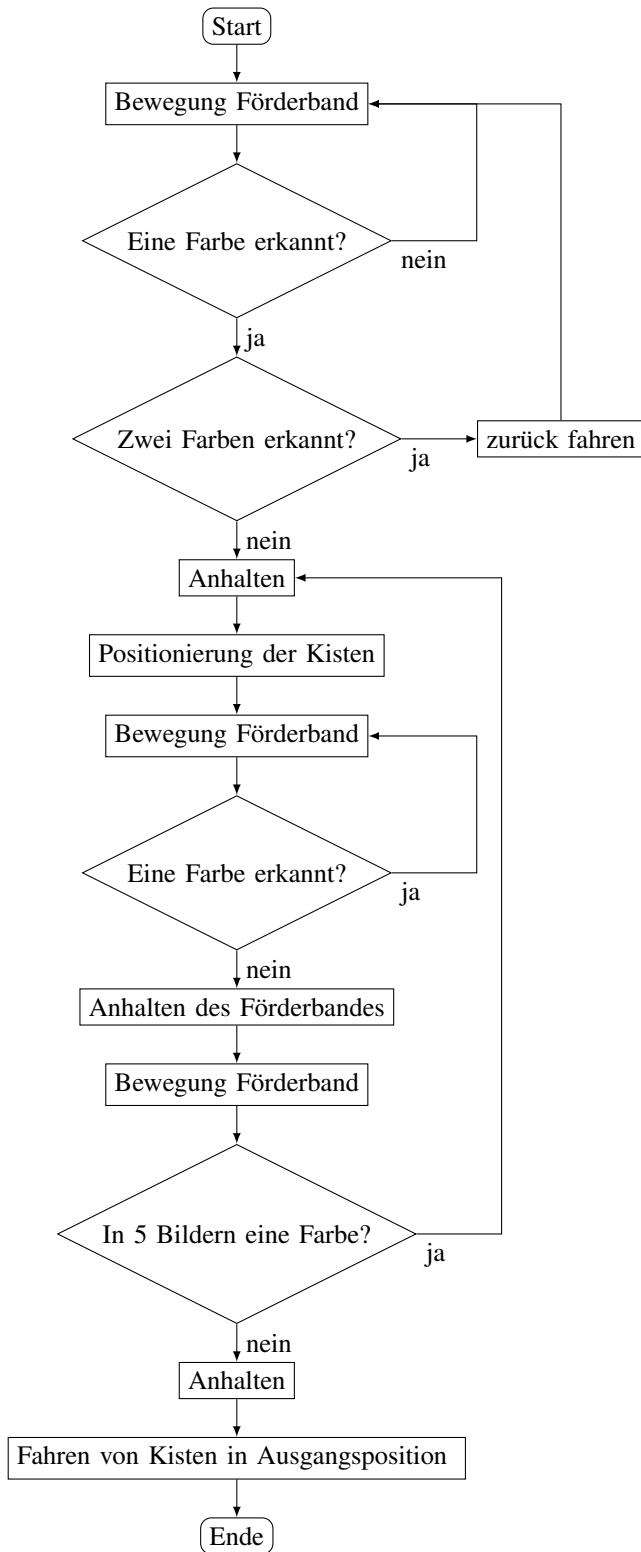


Abbildung 2. Programmablaufplan zur Funktionsweise

B. Aufbau und Funktion der Komponenten

Die Farbsortiermaschine in Abbildung 3 kann man sich aus drei Teilen, bestehend aus zwei hintereinander geschalteten Förderbändern, der Kamera und dem Fahrzeug für die Sortierboxen, vorstellen.

Zwei Förderbänder dienen dem Transport der farbigen Objekte. Damit auch größere Objekte nach ihrer Farbe sortiert werden können, sind zwei Förderbänder über ein Getriebe miteinander verbunden. Als Antrieb für das Band dient ein NXT-Motor, der durch Betätigen des Tastsensors gestartet wird.

Um Bilder der Objekte zu erstellen, wird eine Logitech Webcam verwendet, die extern an den Computer angeschlossen wird und in MATLAB integriert werden kann. Die Bilder der Kamera sind auf die sehr kurze Distanz nicht scharf, müssen es aber auch nicht sein, da es ausreicht eine Farbe im Bild zu erkennen, nicht aber das Objekt genau zu identifizieren.

Die Verwendung eines LEGO-Mindstorms-Farbsensors anstelle einer Kamera wäre nicht möglich gewesen, da die Objekte sehr nah an den Sensor gehalten werden müssten, um eine Farbe zu erkennen.

Damit die Objekte auch sortiert werden können, befindet sich nach dem Förderband ein Fahrzeug, an dem die Sortierboxen befestigt sind. Dieser fährt je nach erkannter Farbe die richtige Box vor das Förderband. Damit die exakte Positionierung der Boxen auch reibungslos funktioniert, ist zusätzlich ein Ultraschallsensor angebracht. Dieser misst permanent den Abstand zum Fahrzeug. Durch die Kopplung des Ultraschallsensors mit dem Fahrzeugmotor können die Kästen punktgenau vor das Förderband gefahren werden.

Da jedoch die Messergebnisse des Ultraschallsensors auf kurze Distanzen teilweise sehr ungenau sind, ist neben dem Sensor noch ein weiterer Tastsensor installiert, der als Anschlagpunkt dient. Fährt das Fahrzeug gegen den Tastsensor, kommt es zum Stillstand.

Damit aber alle verbauten Komponenten auch zusammenarbeiten können, müssen sie miteinander verbunden werden. Dies geschieht über den NXT-Stein, der als zentrale Steuereinheit dient. An ihn werden nicht nur alle Sensoren und Motoren angeschlossen, sondern er stellt auch die Schnittstelle zum MATLAB-Programm dar, mit dem die Komponenten programmiert werden.

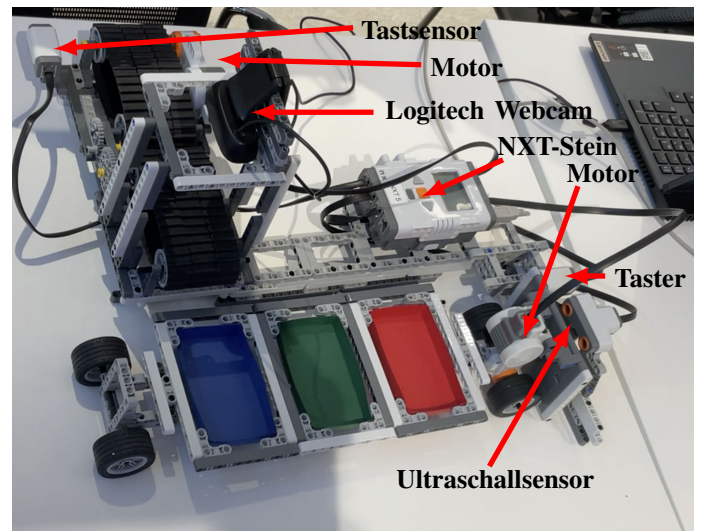


Abbildung 3. Aufbau

C. Wie bekommt man aus einem Bild eine Farbe?

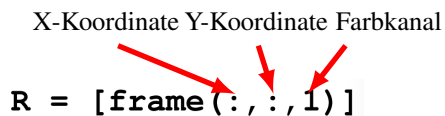
Nachdem der Aufbau und die Idee der Funktionsweise erläutert wurden, besteht die Aufgabe nun darin, mit der Kamera Bilder aufzunehmen und in diesen Farbbereiche zu markieren sowie Aktionen abzurufen, wenn ausgewählte Farben erkannt werden.

```
camera = videoinput('winvideo')
frame = getsnapshot(camera);
image (frame);
R = [frame(:, :, 1)];
G = [frame(:, :, 2)];
B = [frame(:, :, 3)];
```

(Codebeispiel zum Erstellen von Bildern und Auslesen von Farbwerten)

1) *Bild mit einer Kamera erstellen:* Mit dem abgebildeten Code kann eine Kamera in MATLAB eingebunden werden. Dazu wird zunächst ein neues Kameraobjekt mit dem Namen „camera“ initialisiert. Mit den weiteren Befehlen kann dann ein Bild mit der Kamera erstellt und in MATLAB angezeigt werden. Nachdem ein Frame erzeugt wurde, kann aus diesem mit dem Frame-Befehl ein Farbwert ausgelesen werden.

X-Koordinate Y-Koordinate Farbkanal



```
R = [frame(:, :, 1)]
```

Abbildung 4. Aufbau des Frame-Befehls zum Auslesen roter Farbwerte

2) *Farbwert aus einem Bild auslesen:* In Matlab wird ein Koordinatensystem über das Bild gelegt, so dass mit Hilfe der Frame Funktion für jeden Bildpunkt ein Farbwert ausgelesen werden kann.

Der Frame-Befehl (siehe Abbildung 4) besteht aus drei Komponenten. An erster Stelle steht die X-Koordinate, an zweiter Stelle die Y-Koordinate im Bild und an dritter Stelle der Farbkanal (1 rot, 2 grün, 3 blau). Da hier keine konkreten X- und Y-Werte angegeben sind, werden die Farbwerte des gesamten Bildes ausgelesen und in einer Matrix mit dem Namen R gespeichert.

```
mask = (frame(:, :, 1) >= 160)
      & (frame(:, :, 1) <= 255)
      & (frame(:, :, 2) <= 55) &
      (frame(:, :, 3) <= 55);
if any(mask(:)) r=1;
end
```

(Codebeispiel zur Markierung von roten Farbbereichen)

3) *Farbbereiche in einem Bild markieren:* Um nun Farbbereiche in einem Bild markieren zu können, wird die Maskenfunktion verwendet (siehe Codebeispiel zur Markierung von roten Farbbereichen). Dabei werden die Farbwerte mit Hilfe der Frame-Funktion aus dem gesamten Bild ausgelesen. Damit aber nur vordefinierte Farben wie Rottöne, Grüntöne und Blautöne im Bild erkannt werden, muss zuvor über einen RGB-Farbmischer für jede Farbe ein Farbbereich aus der Mischung aller drei Farbwerte (Rot-, Grün-, und Blauwert)

erzeugt werden. Diese mit dem Farbmischer definierten Werte werden verwendet, um Farbbereiche zu definieren, so dass nur Farbwerte in diesem Bereich von der Frame-Funktion ausgelesen werden. Liegen alle drei Farbwerte des Pixels in dem vordefinierten Bereich für Rot-, Grün- oder Blauwert, so wird dieses Pixel durch die „Mask-Funktion“ mit dem Wert 1 markiert. Wenn die Werte außerhalb des Bereichs liegen, wird dem Pixel der Wert Null zugewiesen. Alle Einsen und Nullen werden in einer neuen Matrix gespeichert.

Um zu prüfen, ob eine bestimmte Farbe im Bild vorhanden ist, wird die if Anweisung (siehe Codebeispiel zur Markierung von roten Farbbereichen) verwendet. Dabei wird geprüft, ob in der neuen Matrix Einsen und Nullen vorkommen. Eine Eins in der Matrix reicht aus, um eine Farbe im Bild zu erkennen.

D. Schnelle Verarbeitung von Bildern

Nach dem Aufnehmen von Bildern und dem Erkennen von Farben in einem Bild, geht es nun darum, kontinuierlich und schnell Bilder aufzunehmen und bei jedem neuen Bild zu prüfen, ob die gewünschte Farbe im Bild vorhanden ist.

```
triggerconfig(camera, 'manual');
while
(Bedingung) frame =
    getsnapshot(camera);
end
```

(Codebeispiel zur kontinuierlichen Erzeugung von Bildern)

Der obige Code ermöglicht eine schnelle Erzeugung und die anschließende schnelle Verarbeitung der Daten. Die while-Schleife erzeugt Bilder, solange eine bestimmte Bedingung erfüllt ist. Würde man zur Bilderzeugung nur die while-Schleife verwenden, müsste bei jedem neuen Durchlauf der Schleife das Kameraobjekt geöffnet, eine Erfassung gestartet, ein Frame erstellt, die Erfassung beendet und das Kameraobjekt geschlossen werden. Hinzu kommt noch die Datenverarbeitung. Durch die zeitaufwendigen Schritte entsteht so nur ca. 1 Bild in 3 Sekunden, was viel zu langsam ist, um die Farben der Objekte rechtzeitig zu erkennen und dann sortieren zu können. Mit dem Befehl „triggerconfig (camera, 'manual')“ wird die Verarbeitung weniger zeitintensiv, da nur einmal eine Verbindung zum Gerät hergestellt und eine Konfiguration durchgeführt wird. Man kann sich vorstellen, dass die Kamera im Bereitschaftsmodus gehalten wird. Außerdem werden die Daten nur einmal gespeichert, was eine schnellere Verarbeitung der Bilder in MATLAB ermöglicht. Durch die Verwendung des „Triggermodus“ ist es möglich, 1 Bild in ca. 1 Sekunde zu erzeugen. Erst dadurch wird eine schnelle Farberkennung mit einer Kamera möglich gemacht.

E. Herausforderungen

Bei der Entwicklung der Maschine waren nicht nur konstruktive, sondern auch programmiertechnische Herausforderungen zu bewältigen. Dazu gehörte zunächst die konstruktive Schwierigkeit, die beiden Förderbänder über ein Getriebe miteinander zu verbinden, da die Zahnräder oft nicht die gewünschte Größe aufwiesen.

Das programmiertechnische Problem, das es zu lösen galt, bestand zum einen darin, Farbbereiche in einem Kamerabild zu erkennen und zu markieren, so dass bei Erkennen einer bestimmten Farbe eine ganz bestimmte Aktion ausgeführt wird. Zum anderen war die Interpretation der Messdaten des Ultraschallsensors und die Ansteuerung des Motors, zur exakten Positionierung der Kisten eine kleine Herausforderung. Erschwert wurde dies durch die teilweise ungenaue Messung des Ultraschallsensors auf kurze Distanzen, was durch einen nachträglich eingebauten Tastsensor behoben werden konnte.

IV. ERGEBNISDISKUSSION

Nachdem alle bestandenen Herausforderungen und Probleme fast gelöst sind, kann sich das Endergebnis sehen lassen. Die Farben der Objekte darunter verschiedene Blau-, Grün- und Rottöne, werden in den Kamerabildern schnell erkannt und die Objekte entsprechend ihrer Farbe in Boxen sortiert. Auch mehrfarbige Objekte werden erkannt und aussortiert, indem das Förderband rückwärts läuft. Problematisch ist nur, dass das Aussortieren von mehrfarbigen Objekten nur zu Beginn des Sortiervorgangs funktioniert. Das bedeutet, wenn nach einem Sortiervorgang wieder mehrfarbige Körper auf das Förderband gelegt werden, werden diese nicht mehr erkannt und somit auch nicht mehr aussortiert. Des Weiteren besteht die Problematik, dass teilweise bei mehrfarbigen Körpern nur eine vordefinierte Farbe erkannt wird und somit diese Körper nach ihrer erkannten Farbe sortiert werden, was jedoch einen Sortierfehler darstellt, da nur rein farbige Objekte sortiert werden sollen. Nicht nur bei der Farberkennung können Fehler auftreten, sondern auch bei der Bilderzeugung. Da die Kamera nur ein Bild pro Sekunde aufnimmt, kann es vorkommen, dass ein Bild aufgenommen wird, bevor sich das Objekt im Bildbereich der Kamera befindet und erst wieder, wenn es den Bildbereich verlassen hat. Dies kann dazu führen, dass die Farben der Objekte nicht erkannt werden und ein Sortiervorgang nicht gestartet wird.

V. ZUSAMMENFASSUNG UND FAZIT

Das Ergebnis des zweiwöchigen Praktikums war eine Sortiermaschine, die verschiedene Objekte nach ihrer Farbe (blau, grün und rot) sortieren kann.

In Zukunft könnte an einigen Stellen noch eine Optimierung erfolgen. Zum Beispiel könnte der Code zur Farberkennung in MATLAB noch verkürzt und überarbeitet werden, um so noch schneller und effektiver Farben in einem Bild erkennen zu können. Des Weiteren könnte die Farberkennung mehrfarbiger Objekte erweitert werden, indem der Farbanteil der Farben am Objekt berechnet wird, so dass erst ab einem bestimmten Farbanteil sortiert wird und es nicht zu einer falschen Sortierung von mehrfarbigen Objekten kommt. Auch könnten noch weitere Farben hinzugefügt und die Anzahl der Sortierboxen erweitert werden.

Letztendlich wurde aber das Ziel, die Idee einer Farbsortiermaschine umzusetzen, erreicht. Somit war das LEGO-Praktikum ein voller Erfolg und hat zu dem auch viel Spaß gemacht.

Während des Praktikums konnte man nicht nur seiner Kreativität praktisch freien Lauf lassen, sondern lernte gleichzeitig die Grundlagen, einer neuen und in naturwissenschaftlichen

Bereichen wichtige Programmiersprache, kennen.

In diesem Bericht wird die gesamte Arbeit, der zwei Wochen von der Entwicklung und den konstruktiven Ergebnissen über die Funktionsweise bis hin zu den wichtigsten Programmierschritten der Farbsortiermaschine dargestellt und näher erläutert.

LITERATURVERZEICHNIS

- [1] MathWorks:*triggerconfig*, https://de.mathworks.com/help/imaq/imaqdevice.triggerconfig.html?s_tid=srchtitle_triggerconfig_1 (Stand: Februar 2023)
- [2] MathWorks:*Acquiring a Single Image in a Loop*, <https://de.mathworks.com/help/imaq/acquiring-a-single-image-in-a-loop.html> (Stand:Februar 2023)
- [3] Prasadi Gunawardana:*Geschichte der Roboter* (31.07.2022), <https://wandelbots.com/de/geschichte-der-roboter> (Stand:Februar 2023)
- [4] Amazone:*Logitech-C270-Webcam* <https://www.amazon.de/Logitech-C270-Webcam-720p-schwarz/dp/B01BGBJ8Y0>
- [5] Amazone:*LEGO NXT Ultraschallsensor Ultrasonic sensor* <https://www.amazon.de/LEGO-NXT-Ultraschallsensor-Ultrasonic-sensor/dp/B000PM8180>
- [6] Joachim Boné:*NXT-Stein* <https://www.brick-shop.de/Elektrik-4951.html?language=de>
- [7] LEGO:*Tastsensor* <https://www.amazon.de/Logitech-C270-Webcam-720p-schwarz/dp/B01BGBJ8Y0>
- [8] Brick Owl:*LEGO-Mindstorms EV3 Groß Motor* <https://www.brickowl.de/catalog/lego-ev3-large-servo-motor-set-45502>