DIPLOMARBEIT

Positionsregelung eines Elektromotors mit Hilfe von registrierten Bildpaaren

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus Vincze
und Betreuung durch
Dipl.-Ing. Christoph Nowak und Gerardus Croonen, MSc.

Institut für elektrische Automatisierungs- und Regelungstechnik
Gußhausstraße 25-29/E376, A-1040 Wien

Eingereicht an der Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Bakk.techn. Otto Brechelmacher
Matrikelnummer: 0026002
Pragerstrasse 8
2000 Stockerau

Zuerst möchte ich mich bei meinen Eltern bedanken, die es mir ermöglicht haben das Studium zu absolvieren.

Ein besonderer Dank gilt meinen Betreuern Christoph Nowak und Gerardus Croonen für das interessante Thema, die Unterstützung und die Anregungen bei auftretenden Problemen.

Bei Thomas Spielvogel bedanke ich mich für die Korrektur der schriftlichen Arbeit.

Kurzfassung

Diese Diplomarbeit beschäftigt sich mit der Entwicklung und dem Aufbau eines Demonstratorsystems, wobei die Steuerung für eine Positionierungsaufgabe über eine Videokamera und zwei Elektromotoren läuft.
Es wird ein einfacher, per Hand auf Rädern verschiebbarer Versuchsaufbau errichtet, auf welchem eine senkrecht zum Boden blickende Kamera und zwei Servomotoren montiert werden. An den Achsen der Servomotoren werden kurze Ausleger angebaut, an welchen Filzstifte befestigt sind, die beim Verschieben des Versuchsaufbaus Linien auf den Boden zeichnen. Die Filzstifte liegen nicht im Sichtbereich der Kamera, weshalb sich das geplante System von einem klassischen Visual Servoing Ansatz unterscheidet.
Am Boden vorgezeichnete zweidimensionale Objekte sollen erkannt und umschrieben werden, während der Versuchaufbau verschoben wird. Dadurch ist eine Objektverfolgung außerhalb des Sichtbereichs der Kamera erforderlich. Die dafür notwendigen Bildverarbeitungsschritte sind eine Binarisierung des Eingangsbildes, eine Segmentierung sowie eine Klassifizierung der Objekte. Danach wird der Verschiebungsvektor zwischen zwei aufeinanderfolgenden Bildern berechnet und dient zur Positionsberechnung der Objekte außerhalb des Kamerabildes. Damit die Berechnung möglichst genau ist, ist eine hohe Bildrate erforderlich. Aus diesem Grund wurde die Software auf mehrere Threads aufgeteilt.
Befindet sich ein Objekt im Aktionsradius der Ausleger, so wird es umfahren. Für die Steuerung und das Testen des Systems wurde eine Bedienoberfläche implementiert.
Nach einer ausgiebigen Testphase waren die Resultate der Ausweichbewegungen bei unterschiedlichen Fahr- und Positioniergeschwindigkeiten zufriedenstellend. Die Objektpositionen werden genau berechnet und die Objekte fast ausnahmslos umfahren.

Abstract

This master thesis is concerned with the development and assembly of a demonstration system, of which the control system for the positioning problem consists a video camera and two servo motors.
Therefore a manually moveable construction was constructed, on which the camera is installed perpendicular to the floor plane and the servo motors are mounted.
Short brackets are installed on the axis of the servo motors, to which a marker pen can be fixed, to draw lines on the floor by moving the construction. Contrary to the visual servoing task, the brackets are outside of the observed area of the camera.
As the construction is moved, spots, which are painted on the floor, pass through the image. This spots should be detected and circumscribed. To achieve this, an object tracking outside the camera view is necessary. The required image processing steps are the creation of a binary image out of the input image, the segmentation and the classification of the objects. After that the vector between two subsequent pictures has to be calculated, to figure out the position of the objects outside of the camera view. To get an exact vector of the displacement between two pictures, a high frame rate is necessary. Therefore the software is split into several threads.
If an object reaches the operating range of the brackets, it gets circumscribed. For control and development of the demonstration system an operating interface has been implemented.
After a long testing phase the results of the evasive movements at different drive and position speeds are satisfying. The position of the objects is exactly calculated and the booms drive around the objects successfully most oft the time.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1. Einführung

Bei dieser Diplomarbeit geht es im Wesentlichen darum, in einem Kamerabild Objekte zu erkennen und auf Grund ihrer Position Servomotoren zu steuern. An den Servomotoren werden Ausleger zur Aufnahme von Filzstiften befestigt und ermöglichen es so, um die Objekte herum zu fahren und die dabei getätigten Ausweichbewegungen sichtbar zu machen. Anders als bei dem klassischen Visual Servoing Ansatz befinden sich weder die Servomotoren, noch die daran befestigten Ausleger im Sichtbereich der Kamera.
Diese Aufgabenstellung erfordert ein Zusammenwirken von Computervision und der Steuerung von Servomotoren. Im ersten Schritt erfolgt eine Objekterkennung, welche sich aus mehreren Algorithmen zusammensetzt. Die Anforderungen des Demonstratorsystems bestimmten die Auswahl der implementierten Algorithmen. Einerseits erfordert der verschiebbare Versuchsaufbau, dass auch bei leichten Änderungen der Lichtverhältnisse gute Ergebnisse erzielt werden, andererseits ist eine hohe Bildrate erforderlich, wodurch die einzelnen Algorithmen nicht rechenzeitintensiv ausfallen sollten. Letztere ist für die Bildregistrierung notwendig, damit zwischen zwei aufeinanderfolgenden Bildern keine zu große örtliche Verschiebung stattfindet. Mit der Bildregistrierung ist es möglich Objekte auch außerhalb des Bildbereichs der Kamera zu verfolgen, da ihre Positionen mit Hilfe der Verschiebung zwischen zwei aufeinander folgenden Bildern berechnet werden können.
Somit kennt man die Position einzelner Objekte, ohne sie direkt im Kamerabild zu sehen. Dadurch ist das Ansteuern beziehungsweise das Umfahren der Objektpositionen außerhalb des Sichtbereichs der Kamera möglich.
Dabei kann man sich die Weiterberechnung der Objektpositionen wie ein Überlappen mehrere aufeinander folgender Bilder vorstellen. Aus ihnen entsteht ein Bild, welches von dem aktuellen Kamerabild ausgehend alle Objekte beinhaltet, die sich bis zum Ende des Wirkungsbereiches der Motoren befinden. Aus diesem errechneten Bild werden dann die Positionen für den Ausweichvorgang der Servomotoren ermittelt.
Die Positionierung außerhalb des Bildbereichs der Kamera kann überall dort eingesetzt werden, wo es wegen des notwendigen Platzes oder aus Gründen der Sicherheit nicht möglich ist im Sichtbereich zu agieren oder aber eine zu aufwendige Konstruktion dafür notwendig wäre. Vor allem für die Positionieraufgabe mit großen Werkzeugen, welche entsprechend große Führungen, Halterungen und Antriebe benötigen, bietet sich diese Technik an.

1.2. Ergebnisse

Testfahrten mit dem Demonstratorsystem zeigten, dass das Zusammenspiel der Bildverarbeitungssoftware und der Motorsteuerung für die gegeben Aufgabenstellung gut funktioniert. Das System kann bei unterschiedlichen Positionier- und Fahrgeschwindigkeiten eingesetzt werden und liefert brauchbare Ergebnisse. Lediglich starke Änderungen der Lichtverhältnisse beeinflussen die Bildverarbeitung nachhaltig, so dass keine oder die falschen Objekte erkannt werden und dadurch die Steuerung nicht mehr einwandfrei funktioniert.
Die Geometrie der Ausleger für die Positionierung hat einen starken Einfluss auf die Ausweichbewegung. Ebenso wird dadurch der minimale Abstand bestimmt, den zwei Objekte voneinander entfernt sein müssen, damit die Ausleger zwischen zwei Objekten in die Bildmitte zurückkehren können.
Obwohl der Bildregistrierung nur einige wenige Objekte für die Berechnung des Verschiebungsvektors zur Verfügung stehen, ist eine exakte Positionierung möglich.

1.3. Aufbau der Arbeit

Das erste Kapitel liefert eine Einführung und eine strukturelle Gliederung der vorliegenden Diplomarbeit. Das nachfolgende Unterkapitel 1.4 beschreibt die konkrete Aufgabenstellung.
Im zweiten Kapitel werden die theoretischen Grundlagen und Abarbeitungsschritte für eine Objekterkennung in der Bilderverarbeitung erläutert und die Grundlagen für eine Bildregistrierung erklärt.
Das dritte Kapitel beschreibt den Aufbau des Demonstratorsystems. Hier erfolgt eine Beschreibung der einzelnen mechanischen Komponenten des Versuchsaufbaus, sowie der verwendeten Hardware.
Im nachfolgenden Kapitel wird die Steuerung der Servomotoren mittels der Frequenzumrichter präsentiert. Zuerst erfolgt eine Einführung über die Steuerung mit der Software des Herstellers und im Anschluss daran wird die Softwareimplementierung zur Steuerung der Servomotoren mittels der Programmiersprache C# erläutert. Es werden die einzelnen Funktionen für die Kommunikation über Ethernet und das Ändern von Parametern der Frequenzumrichter vorgestellt.
Das fünfte Kapitel beschäftigt sich mit der Implementierung der einzelnen Bildverarbeitungsschritte und dem Auslösen des Ausweichvorganges. Somit stellt es die Verbindung zwischen dem Bildverarbeitungssystem und der Motorsteuerung dar. Ebenso wird auf die Berechnung der Positionen der Servomotoren eingegangen und die Implementierung in mehreren Threads erläutert.
Das sechste Kapitel präsentiert die Bedienoberfläche des Demonstratorsystems, welche für das Testen der Steuerung entwickelt wurde. Hier erfolgt eine Beschreibung der einzelnen Funktionen und Parameter der Testumgebung.
Das siebente Kapitel beinhaltet die Vermessung und Ausrichtung des Demonstratorsystems.
Im achten Kapitel werden die Ergebnisse präsentiert. Es werden Bilder von Ausweichvorgängen mit unterschiedlichen Positionier- und Fahrgeschwindigkeiten sowie Betriebsmodi gezeigt und auf die Besonderheiten eingegangen.
Im neunten und letzten Kapitel werden Probleme des Demonstratorsystems aufgezeigt und Lösungs- beziehungsweise Verbesserungsvorschläge geliefert.

1.4. Aufgabenstellung

Das Ziel ist die Entwicklung eines Demonstratorsystems welches zweidimensionale Objekte erkennt und diesen ausweicht. Dazu ist ein verschiebbarer Versuchsaufbau notwendig, auf welchem eine Videokamera, ein Rechner, sowie Servomotoren und die zugehörigen Frequenzumrichter montiert werden. Um die Objekte umfahren zu können werden auf den Motorachsen kurze Ausleger zur Aufnahme von Filzstiften befestigt. Die Servomotoren sitzen seitlich vor dem Stativ für die Kamera. Die Ausleger werden so montiert, dass die Ausweichbewegungen unterhalb des Versuchaufbaus erfolgen. Jeder Ausleger hat eine Reichweite vom näher gelegenen Bildrand bis kurz vor die Bildmitte, wodurch verhindert wird, dass die beiden Ausleger miteinander kollidieren.
Die Steuerung für die Positionierungsaufgabe erfolgt über eine Videokamera und zwei Servomotoren. Im Gegensatz zum Visual Servoing Ansatz befinden sich die die beiden Motoren außerhalb des Sichtbereiches der Kamera. Das Prinzip ist in Abbildung 1 dargestellt. Daraus ist ersichtlich, dass die Positionen der Objekte ermittelt und über den Sichtbereich der Kamera hinaus weiterberechnet werden müssen, bis sie von den Filzstiften umschrieben werden können. Der Bildbereich der Kamera befindet sich in Fahrtrichtung vor dem Versuchsaufbau und wurde in Abbildung 1 dunkelgrau dargestellt. Damit werden Bilder der befahrenen Strecke und der Objekte, welche sich darauf befinden, aufgenommen. Aus dem Kamerabild werden die Objektpositionen erkannt und anschließend die Steuersignale für das Umfahren der einzelnen Objekte generiert, damit die Ausleger, die an den Achsen der

Servomotoren montiert sind und die Filzstifte aufnehmen, unterhalb des Versuchaufbaus den Objekten ausweichen. Die Kommunikation zwischen dem Bildverarbeitungssystem und den Frequenzumrichtern der Servomotoren erfolgt über Ethernet.

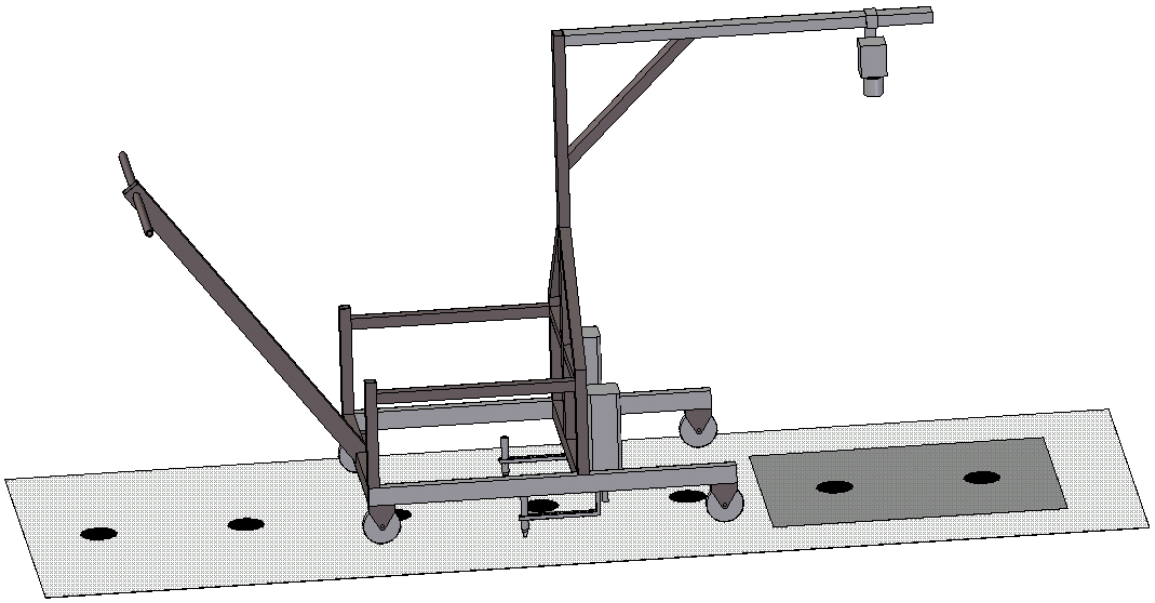


Abbildung 1: Funktionsweise des Demonstratorsystems

Die Entwicklung des Demonstratorsystems wurde in vier Arbeitspakete gegliedert:

1. Versuchsaufbau:
   - Gesamtsoftwarekonzept erstellen
   - Versuchsaufbau entwerfen und aufbauen
   - Anbindung der Kamera über FireWire an den Rechner
   - Ansteuerung der Frequenzumrichter über Ethernet
2. Bildverarbeitungssystem:
   - Objektdetektion
   - Bildregistrierung
3. Servoing

   Berechnung der Position der Objekte außerhalb des Sichtbereiches der Kamera, anhand der Position der detektierten Objekte im Videobild und dem Abstand der Kamera zu den Servomotoren. Daraus sind die Positionen der Ausleger beziehungsweise der Filzstifte und die erforderlichen Ethernet Steuersignale für die Servomotoren abzuleiten, damit durch ein örtliches Verschieben des Versuchsaufbaues die Objekte umfahren werden können.
4. Funktionstests, Evaluierung und Optimierung

Die Software des Demonstratorsystems setzt sich im Wesentlichen aus drei Teilen zusammen. Der erste Teil ist die Bildverarbeitung, wo die Bildaufnahme erfolgt und die Objekterkennung stattfindet. In einem weiteren Schritt muss die Position der Objekte im Bild bestimmt werden und auf Grund der Bewegung des Versuchsaufbaus deren Position außerhalb dem

Sichtbereich der Kamera weiterberechnet werden. Hierzu ist es erforderlich die Verschiebung zweier aufeinanderfolgender Bilder zu ermitteln. Ein weiterer Schritt ist die Berechnung der für das Umfahren der Objekte notwendigen Positionen der Servomotoren aus den Objektkoordinaten.
Der zweite Teil der Software ist für die Steuerung der Servomotoren zuständig. Dieser beinhaltet Funktionen zum Verbindungsaufbau und zum Senden der Daten an die Frequenzumrichtern sowie Funktionen zum Erstellen der für die Ethernetkommunikation benötigten Datenrahmen. Des Weiteren ist die Implementierung von Funktionen notwendig welche Parameterwerte der Frequenzumrichter verändern können, damit mit einem Funktionsaufruf die Ausweichbewegung der Ausleger ausgeführt werden kann. Es sollen zwei Formen von Ausweichbewegungen möglich sein. Einerseits wo sich die Ausleger in der Bildmitte befinden, nur vor einem Objekt öffnen und danach wieder schließen. Für den Fall, dass sich das Objekt nicht in der Bildmitte befindet, wird die Ausweichbewegung nur von einem Servomotor durchgeführt und der andere Ausleger verweilt in der Bildmitte. Alternativ dazu sollen die Ausleger zwischen dem verlängerten Bildrand und der Bildmitte hin- und herschwingen. Für den Fall, dass sie einem Objekt ausweichen müssen, wird die Bewegung um das Objekt herum weitergeführt. So wie bei dem oben beschrieben Ausweichvorgang erfolgt das Ausweichen von Objekten in der Bildmitte mit beiden Servomotoren und seitlich durch den näher liegenden Servomotor, während der andere die Schwingbewegung bis zur Bildmitte weiterführt.
Der letzte Teil der Software ist die Implementierung einer Bedienoberfläche zum Betreiben und Testen des Demonstratorsystems. Dadurch soll es möglich sein die Steuerung für verschiedene Geschwindigkeiten und Objektgrößen zu testen. Ebenso soll ein Wechseln der beiden Betriebsmodi möglich sein und die Speicherung von Ergebnisbildern ist für die Testphase und das Überprüfen der Algorithmen erforderlich.
Da von der Bildaufnahme bis zur Objektverfolgung mehrere teilweise sehr rechenzeitintensive Verarbeitungsschritte notwendig sind, wird die Bildverarbeitungssoftware in mehreren Threads realisiert, welche auf die einzelnen Prozessorkerne aufgeteilt werden können. Dadurch ist es möglich bei der Objektverfolgung mehrere Bilder pro Sekunde zu verarbeiten. Dies hat den Vorteil, dass die Verschiebung zweier aufeinanderfolgender Bilder relativ klein ausfällt und die Berechnung der Objektposition genauer wird. Aus diesem Grund ist es auch möglich mehrere Objekte pro Bild zu bearbeiten, da die Bewegung besser verfolgt werden kann. Ist der Abstand zwischen zwei Objekten kleiner als die Verschiebung zweier Bilder zueinander, können die Objekte einander nicht mehr eindeutig zugeordnet werden und eine Objektverfolgung ist somit unmöglich.

2. Theoretische Grundlagen

Die Bildverarbeitung im industriellen Bereich benötigt für die Regelung und Steuerung von Prozessen oder das Überprüfen von Werkstücken Informationen aus dem Kamerabild. Hierzu ist eine Reihe von Verarbeitungsschritten notwendig, bis aus dem von der Kamera aufgenommenen Bild die für die Anwendung notwendigen Informationen extrahiert werden können. Diese lassen sich nach [2] in drei Schritte gliedern:

1. Aufnahme und Vorverarbeitung der Bilddaten
2. Merkmalsextraktion
3. Bildanalyse

Der erste Verarbeitungsschritt betrifft die Errichtung der für die Aufnahme notwendigen Bedingungen. Hierzu zählen unter anderem die Art und Form der Beleuchtung, die Wahl des bildgebenden Verfahrens oder aber die Ausrichtung der Kamera beziehungsweise des Werkstückes. Weiters beinhaltet er auch die Kalibrierung der Kamera und die Digitalisierung des Bildes.
Bei dem zweiten Verarbeitungsschritt, der Merkmalsextraktion, wird versucht einzelne wichtige Merkmale der aufgenommenen Szene für das Bildverarbeitungssystem sichtbar oder deutlicher zu machen. Hierzu werden Verfahren wie Glättung, Binarisieren oder Kantendetektion verwendet.
Als letzter Schritt folgt die Bildanalyse bei der aus den gewonnenen Merkmalen beispielsweise einzelne Bildregionen segmentiert werden. Ebenso werden morphologische Operationen durchführt oder es erfolgt die Klassifizierung von gefundenen Objekten.

2.1. Berechnung eines Grauwertbildes aus einem Farbbild

Für viele technische Anwendung stehen unterschiedliche Formen von Kameras zur Verfügung. Farbkameras bieten die Möglichkeit bei der Bildverarbeitung auch Farbfilter zu verwenden.
Werden jedoch keine Farbinformationen benötigt, so empfiehlt sich eine Umwandlung der Farbbilder in Grauwertbilder. Dies bietet den Vorteil, dass Grauwertbilder deutlich platzsparender als Farbbilder gespeichert werden können, da nicht mehr drei Werte für die Farben oder den Farbton, Sättigung und Dunkelstufe gespeichert werden müssen sondern lediglich der Grauwert des entsprechenden Bildpunktes.
Ebenso bietet sich die Möglichkeit, die Grauwerte der einzelnen Bildpunkte gleich in eine Matrix zu schreiben welche lediglich ein Feld von ganzen Zahlen darstellt. Dadurch kann, losgelöst vom jeweiligen Bildformat, in weiterer Folge Speicherplatz und Rechenzeit gespart werden. Außerdem gestaltet sich der Zugriff auf ein Feld meist einfacher und Rechenzeitschonender als auf die Elemente eines Bildes.
Der Grauwert g stellt den Helligkeits- oder Intensitätswert eines einzelnen Bildpunktes dar, wobei R für den Rotanteil, G für den Grünanteil und B für den Blauanteil des Bildpunktes steht, und wird wie folgt berechnet:

$$g = 0{,}3 * R + 0{,}59 * G + 0{,}11 * B. \qquad \textbf{(2.1)}$$

2.2. Verfahren von Otsu

Das Verfahren von Otsu wird zur Erzeugung von Binärbildern aus Grauwertbildern benutzt und verwendet zur Berechnung der Schwelle für die Binärbilderzeugung statistische Methoden [11]. Dabei wird die Schwelle so gewählt, dass die Varianz zwischen den beiden Klassen möglichst groß ist.
Da für die Berechnung des Schwellwertes lediglich das Histogramm des Grauwertbildes benötigt wird, welches in einem eindimensionalen Feld gespeichert werden kann, kann dieser sehr schnell mit dem Computer berechnet werden.
Für die Berechnung des Schwellwertes eines Grauwertbildes mit L Intensitätsstufen von 0 bis L-1 und der Größe M x N, bezeichnet n_i die Anzahl der Pixel mit dem Intensitätswert i, welche aus dem Histogramm ausgelesen werden können.
Die Gesamtzahl aller Bildpunkte berechnet sich als

$$M*N = n_0+n_1+\ldots+n_{L-1}. \quad \textbf{(2.2)}$$

Weiters wird ein normiertes Histogramm benötigt, bei dem das Histogramm des Bildes auf die Gesamtzahl aller Bildpunkte bezogen wird.

$$p_i = \frac{n_i}{M * N}. \quad \textbf{(2.3)}$$

Für die einzelne Komponenten p_i des normierten Histogramms gilt:

$$\sum_{i=0}^{L-1} p_i = 1, p_i \geq 0. \quad \textbf{(2.4)}$$

Nun wird eine Schwelle T(k)=k gesucht, die im Bereich 0<k<L-1 liegt und die das Eingangsbild in zwei Klassen C_1 und C_2 teilt. Hierbei enthält die Klasse C_1 alle Bildpunkte mit der Intensität von 0 bis k und die Klasse C_2 alle Bildpunkte mit einem Grauwert von k+1 bis L-1.
Die Wahrscheinlichkeit P_1 (k), welche angibt, ob ein Bildpunkt zur Klasse C_1 gehört, berechnet man mit der kumulativen Summe

$$\mathrm{P_1(k)} = \sum_{i=0}^{k} p_i. \quad \textbf{(2.5)}$$

Ebenso kann die Wahrscheinlichkeit P_2, dass ein Bildpunkt zur Klasse C_2 gehört berechnet werden

$$\mathrm{P_2(k)} = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k). \quad \textbf{(2.6)}$$

Der Mittelwert der Grauwerte der Klasse C_1 berechnet sich als

$$\mathrm{m_1(k)} = \sum_{i=0}^{k} i * P(i/C_1) = \sum_{i=0}^{k} i * \frac{P(C_1/i) * P(i)}{P(C_1)} = \frac{1}{P_1(k)} \sum_{i=0}^{k} i * p_i. \quad \textbf{(2.7)}$$

Ebenso lässt sich die der Mittelwert der Grauwertverteilung für die Klasse C_2 ermitteln

$$\mathrm{m_2(k)} = \sum_{i=k+1}^{L-1} i * P(i/C_2) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} i * p_i. \quad \textbf{(2.8)}$$

Der kumulative Mittelwert m(k) bis zum Schwellwert k ergibt sich als

$$m(k) = \sum_{i=0}^{k} i * p_i. \quad \textbf{(2.9)}$$

Die durchschnittliche Intensität des Grauwertbildes (der globale Mittelwert), berechnet sich als

$$m_G(k) = \sum_{i=0}^{L-1} i * p_i. \quad \textbf{(2.10)}$$

Die globale Varianz σ^2_G beschreibt die Varianz der Grauwerte aller Bildpunkte

$$\sigma_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 * p_i , \quad \textbf{(2.11)}$$

wohingegen σ^2_B die Varianz zwischen den beiden Klassen angibt. Diese berechnet sich als

$$\sigma_B^2(k) = P_1 * (m_1 - m_G)^2 + P_2 * (m_2 - m_G)^2 = \frac{(m_G * P_1(k) - m(k))^2}{P_1(k) * (1 - P_1(k))} . \quad \textbf{(2.12)}$$

Der hintere Teil der Gleichung (2.12) ist für die Schwellwertberechnung mit einem Computer besonders gut geeignet, da der globale Mittelwert m_G für jedes Bild nur einmal berechnet werden muss und folglich nur die beiden Parameter m und P_1 für alle L Grauwerte zu bestimmen sind.
Je weiter die kumulativen Mittelwerte der beiden Klassen m_1 und m_2 auseinander liegen, desto größer ist die Varianz zwischen den beiden Klassen.
Der bestmögliche Schwellwert k* liegt bei

$$\sigma_B^2(k^*) = \max_{0 \le k \le L-1} \sigma_B^2(k^*) , \quad \textbf{(2.13)}$$

wobei die Varianz zwischen den beiden Klassen den größtmöglichen Wert erreicht. Ebenso kann man die Güte η der Schwelle bestimmen

$$\eta = \frac{\sigma_B^2}{\sigma_G^2} , \quad \textbf{(2.14)}$$

beziehungsweise die Schwelle aus einem möglichst großen Wert von η berechnen, da es sich bei σ^2_G um eine Konstante handelt.
Der Vorteil des Verfahrens von Otsu gegenüber einem festen Schwellwert besteht darin, dass auf Helligkeitsänderung oder Änderungen des Lichtes von einem zum nächsten Bild auch die Schwelle für die Binarisierung angepasst wird.
Hat man jedoch einen festen Schwellwert, dann können Änderungen der Beleuchtung oder des Hintergrundes zu einer deutlichen Verschlechterung der Bildqualität führen, was meist auch Informationsverlust zur Folge hat.

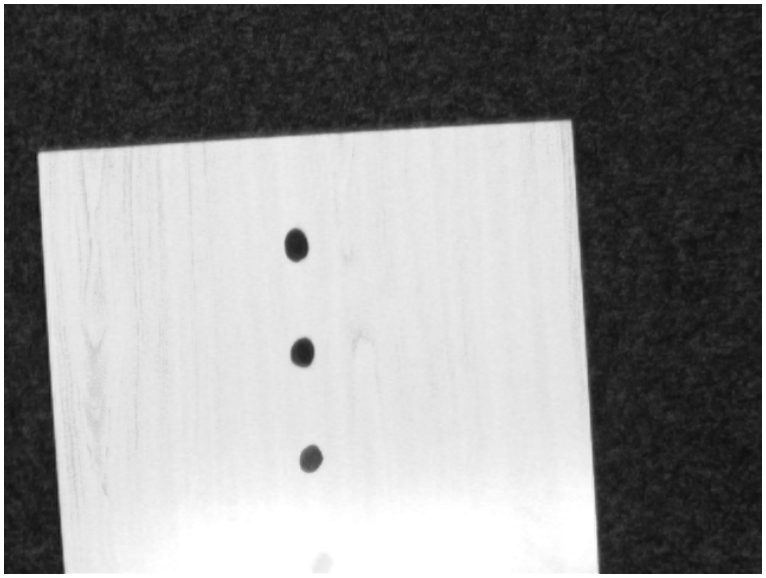


Abbildung 2: Grauwertbild

Da das Graustufenbild aus Abbildung 2 im unteren Teil sehr stark beleuchtet wird, ist der unterste Punkt kaum noch sichtbar. In Abbildung 3 ist das Histogramm des Graustufenbildes dargestellt. Es handelt sich hierbei um ein bimodales Histogramm, das heißt, dass es zwei

voneinander deutlich getrennte Spitzen hat, welche in diesem Fall auch noch relativ weit von einander entfernt sind. Dies würde ausreichend Spielraum für eine fixe Schwelle bieten.

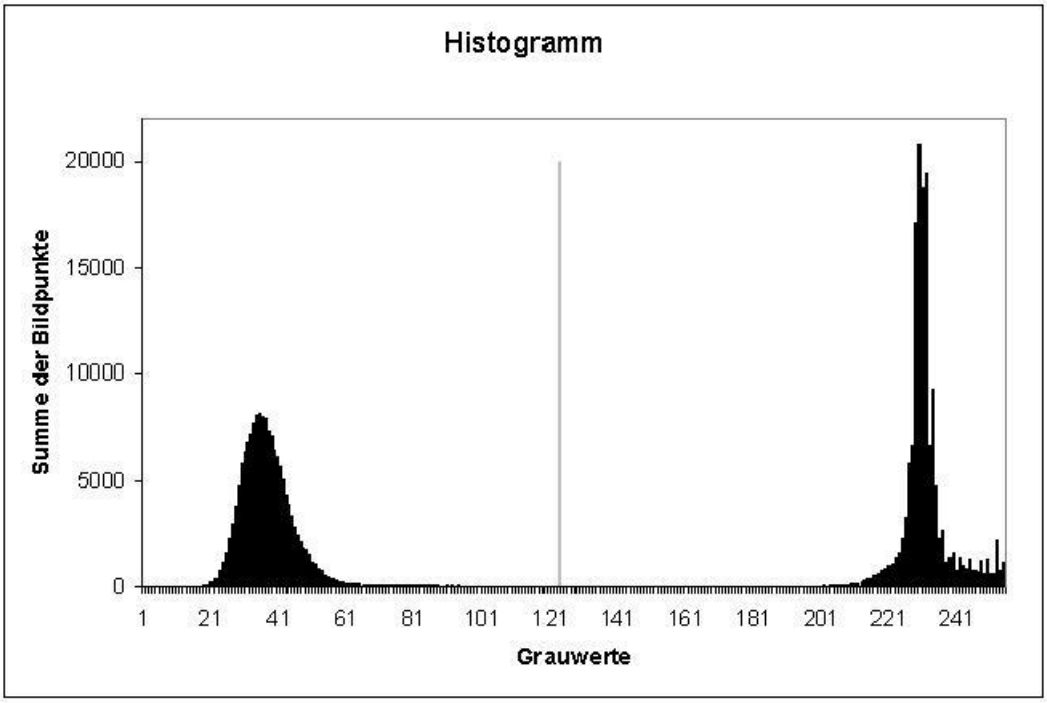


Abbildung 3: Histogramm des Grauwertbildes mit der Schwellen nach dem Verfahren von Otsu (122)

Der Schwellwert nach dem Verfahren von Otsu beträgt für dieses Bild 122 und liefert ein sehr gutes Binärbild (Abbildung 4). Der aufgrund der starken Beleuchtung schon im Graustufenbild schwer sichtbare Punkt ist hier nicht mehr zu sehen, aber auch die Maserung des Holzes und das Muster des Hintergrundes sind nicht mehr zu erkennen. Da es sich bei der Binarisierung eines Bildes um Merkmalsextraktion handelt wurden hier die interessanten Objekte (die dunklen Punkte), von ihrer Umgebung gelöst und stehen nun für weitere Bildverarbeitungsschritte zur Verfügung.

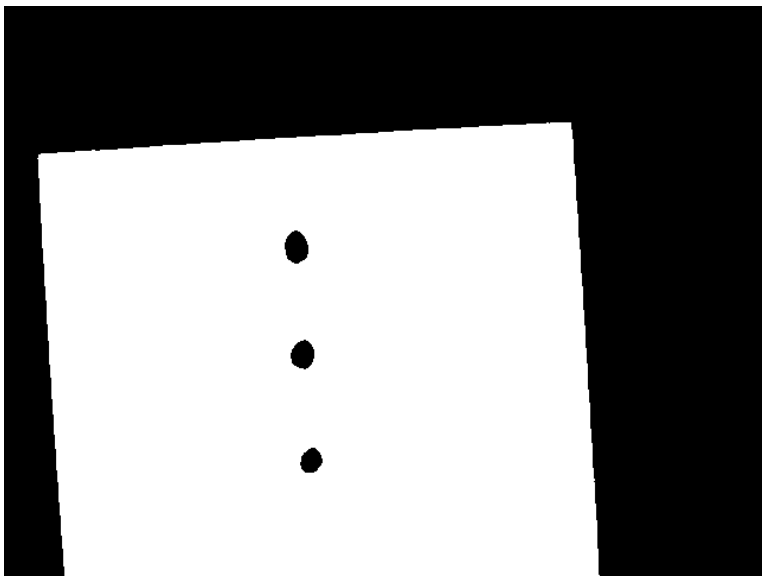


Abbildung 4: Binärbild mit Otsu-Schwelle (122)

In Abbildung 5 ist ein Schwarzweißbild zu sehen, welches mit einem Schwellwert von 50 binarisiert wurde. Wie man dem Histogramm entnehmen kann, befindet sich diese Schwelle in der Nähe des ersten Maximums und führt zu einem schlechteren Resultat als nach dem Verfahren von Otsu. Die Schwelle wird wesentlich stärker zu den dunkleren Grauwerten verschoben, weshalb nur noch sehr dunkle Bildpunkte in schwarze Bildpunkte umgewandelt werden. Nachteilig daran ist, dass jetzt schon die beiden untersten Punkte nicht mehr vorhanden und die oberen beiden Punkte deutlich kleiner sind. Zusätzlich ist der Hintergrund nicht mehr eine geschlossene Fläche, sondern besitzt sehr viele weiße Flächen. Außerdem sind die Kanten nicht mehr so klar wie in Abbildung 4.

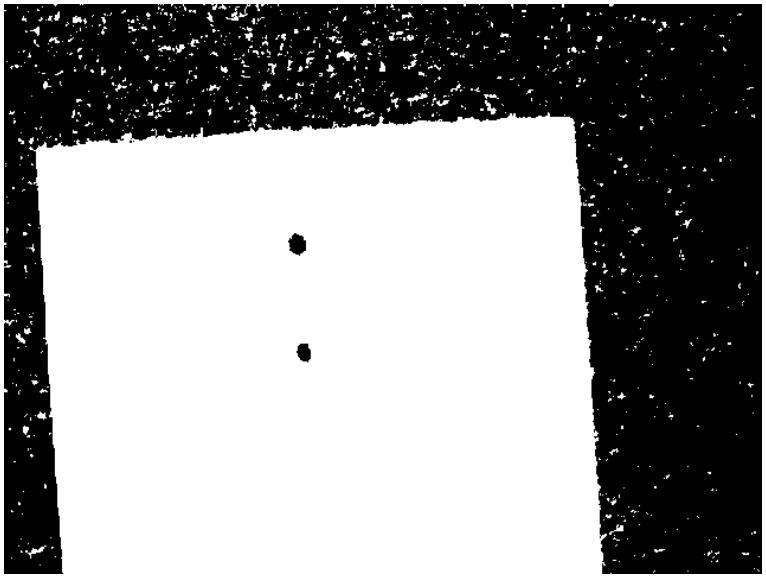


Abbildung 5: Binärbild mit der Schwelle bei Grauwert 50

Eine Verschiebung des Schwellwertes zu dem helleren Grauwerten verschlechtert ebenso das Resultat der Binarisierung, wie in Abbildung 6 zu sehen ist. Es werden nur noch ganz helle Grauwerte als weiß interpretiert, alle anderen als schwarz, sodass selbst die Maserung des Holzes sichtbar wird. Dies hat zur Folge, dass neben den eigentlichen Objekten noch mehr Information im Bild vorhanden ist, welche wieder herausgefiltert werden muss.

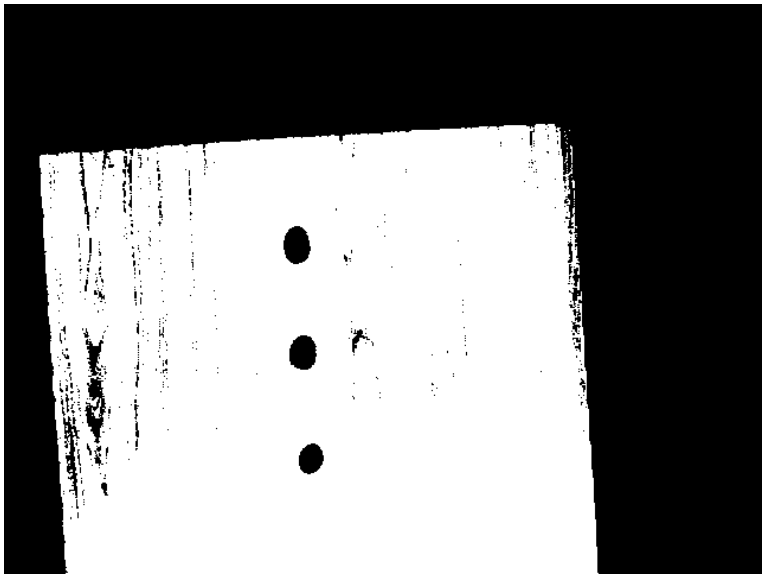


Abbildung 6: Binärbild mit der Schwelle bei Grauwert 220

Für die Weiterverarbeitung des Bildes ist die Wahl der bestmöglichen Schwelle besonders wichtig, da im Gegensatz zu Abbildung 4 in Abbildung 5 und Abbildung 6 bei einer Segmentierung wesentlich mehr Regionen gefunden als benötigt werden, was wiederum einen Mehraufwand bei der Filterung der Segmente zur Folge hat. Dieses Problem könnte man durch den Einsatz von morphologischen Operationen reduzieren, indem man die Anzahl der nicht benötigten Flächen verringert. Die Information des verlorenen Bildpunktes kann man jedoch mit keinem der Verfahren zurückgewinnen. Dies verdeutlicht wie wichtig die richtige Beleuchtung bei der Bildaufnahme ist.
Für Bilder mit einem bimodalen Histogramm deren Maxima aber nicht so weit auseinander liegen, oder wo es nicht nur sehr dunkle und sehr helle Regionen gibt, sondern die Grauwerte breiter gefächert sind, ist die Wahl des richtigen Schwellwertes wichtig, da sonst bei der Binarisierung Informationen verloren gehen wenn Objekte nicht mehr dargestellt werden oder mit dem Hintergrund verschmelzen.

2.3. Segmentierung

Bei der Bildsegmentierung handelt es sich um die Identifikation von Bildregionen. Dabei wird der Bildbereich durchlaufen und nach Haberäcker: *„Bei der Bildsegmentierung wird der Ortsbereich des Bildes, also der Berech der Zeilen- und Spaltenkoordinaten x und y, in Bereiche eingeteilt, die nach Maßgabe eines Einheitlichkeitsprädikats P zusammengehören“*[1]. Diese inhaltlich zusammenhängenden Gebiete werden als Regionen oder Segmente bezeichnet. Im Allgemeinen unterliegen diese Segmente gewissen Regeln [10]:

- Jeder Bildpunkt wird genau einem Segment zugeordnet.
- Wird das Einheitlichkeitsprädikat auf die Bildpunkte eines Segmentes angewendet dann ergibt sich der Wahrheitswert ’wahr’.
- Wird ein Bildpunkt zu einer anderen Region hinzugefügt so ergibt sich der Wahrheitswert ’falsch’.

Ausgehend von einem Binärbild, wird jeder Bildpunkt überprüft und mit einer Markierung versehen, falls es sich um einen schwarzen Bildpunkt handelt. Bildpunkte die eine zusammenhängende Fläche bilden werden mit der gleichen Markierung versehen und stellen somit ein Segment dar.

2.3.1. Component-Labeling Algorithmus

Ein sehr effizienter und schneller Segmentierungsalgorithmus wird in [4] beschrieben. Im Gegensatz zu anderen Algorithmen wird hier das Bild nur ein einziges Mal durchlaufen. Dies wird durch eine Konturverfolgung ermöglicht, welche die äußere und mögliche innere Konturen einer Region erkennt, verfolgt und markiert. Dadurch kommt es vor, dass einzelne Bildpunkte, die sich auf dem Objektrand befinden, auch mehrmals durchlaufen werden.
Der Ausgangspunkt für die Segmentierung ist ein Binärbild, welches von links oben nach rechts unten durchlaufen wird. Wird ein Objekt gefunden, so wird dessen äußere Kontur verfolgt bis man wieder am Startpunkt angelangt ist. Für den Fall, dass Segmente und somit auch deren Kontur am Bildrand liegen, wird ein zusätzlicher weißer Rand um das Bild erzeugt, da ein weißer Randpunkt erforderlich ist um bei der Konturverfolgung die Richtung zu ändern. Alternativ dazu könnte man auch Abfragen implementieren, welche beim Erreichen des Bildrandes die Suchrichtung ändern.

[1] [10] Seite 235

Wird ein schwarzes Pixel P gefunden, welches noch kein Label besitzt und der Bildpunkt oberhalb von P ein weißer Bildpunkt ist (siehe Abbildung 7), so befindet sich P auf der äußeren Kontur eines noch nicht markierten Segments.

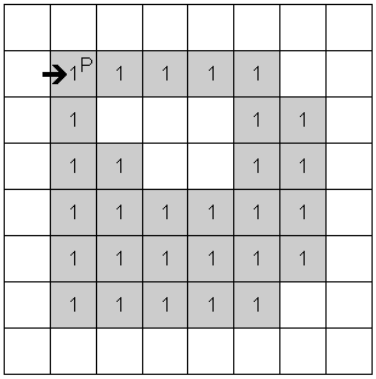


Abbildung 7: Unmarkiertes Segment, Startpunkt der äußeren Kontur (1: nicht markiertes schwarzes Pixel)

Der Punkt P wird mit dem aktuellen Label versehen, danach wird die Konturverfolgung gestartet. Dadurch erhalten alle Bildpunkte die sich auf der äußeren Kontur des Segmentes befinden dasselbe Label wie der Bildpunkt P. Anschließend wird die Labelnummer erhöht und die Suche fortgesetzt.
Befindet sich unterhalb von Bildpunkt P ein nicht markierter weißer Bildpunkt, so liegt dieser auf einer inneren Kontur, siehe Abbildung 8. Hierbei muss man zwei Fälle unterscheiden.
Falls der Bildpunkt P bereits ein Label besitzt so befindet er sich, wie in Abbildung 8 ersichtlich, sowohl auf einer inneren als auch auf einer äußeren Kontur.

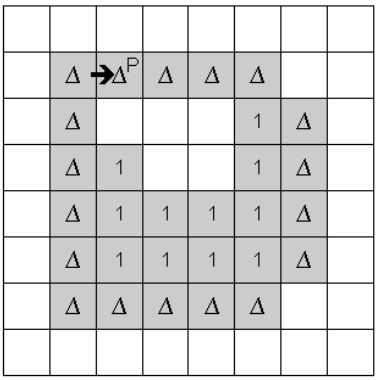


Abbildung 8: P ist der Startpunkt einer inneren Kontur und liegt ebenso auf einer äußeren Kontur (1: nicht markiertes schwarzes Pixel, Δ: markiertes schwarzes Pixel)

Handelt es sich hingegen bei P um ein nicht markiertes schwarzes Pixel, so stellt es der Startpunkt einer neuen inneren Kontur dar und wird mit dem Label des benachbarten Bildpunktes N markiert, wie in Abbildung 9 zu sehen ist.
In beiden Fällen wird die Konturverfolgung gestartet und somit werden alle Randpunkte, sofern keine weitere innere Kontur vorhanden ist, mit demselben Label versehen.

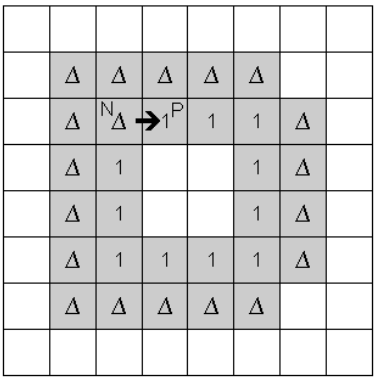


Abbildung 9: P ist der Startpunkt einer inneren Kontur (1: nicht markiertes schwarzes Pixel, Δ: markiertes schwarzes Pixel)

Befindet sich der Bildpunkt P weder auf einer inneren noch auf einer äußeren Kontur, dann wird er beim normalen Bilddurchlauf als unmarkierter schwarzer Bildpunkt der sich innerhalb einer Kontur befindet, erkannt und mit dem Label des Nachbarpunktes N, wie in Abbildung 10 dargestellt, markiert.

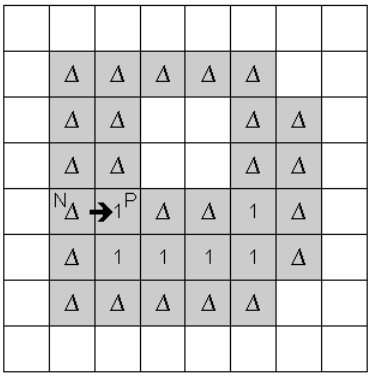


Abbildung 10: P wird mit dem Label des Nachbarpunktes markiert (1: nicht markiertes schwarzes Pixel, Δ: markiertes schwarzes Pixel)

Damit die Konturverfolgung nicht vorzeitig abgebrochen wird, wie beispielsweise am Punkt Q in Abbildung 11, werden die weißen Bildpunkte, welche die Figur umranden mit einem negativen Zahl markiert.

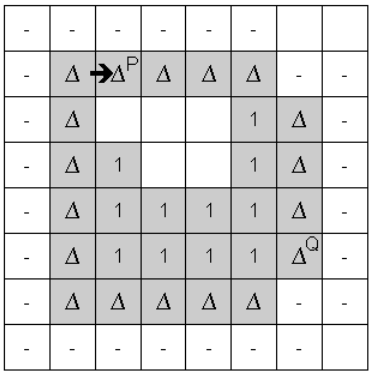


Abbildung 11: Markierung der äußeren Umrandung (weißen Pixel)(1: nicht markiertes schwarzes Pixel, Δ: markiertes schwarzes Pixel, -: weißes Pixel als Umrandung markiert)

Wenn die Konturverfolgung für die äußere Kontur abgeschlossen ist, so befinden sich rund um das Segment markierte weiße Pixel. Ebenso werden bei der Verfolgung der inneren Kontur die weißen Randpixel mit einer negativen Zahl markiert (siehe Abbildung 12).

-	-	-	-	-	-		
-	Δ	Δ	→Δ^R	Δ	Δ	-	-
-	Δ	-	-	-	Δ	Δ	-
-	Δ	Δ	-	-	Δ	Δ	-
-	Δ	1	Δ	Δ	1	Δ	-
-	Δ	1	1	1	1	Δ	-
-	Δ	Δ	Δ	Δ	Δ	-	-
-	-	-	-	-	-	-	

Abbildung 12: Markierung der inneren Umrandung (weißen Pixel)(1: nicht markiertes schwarzes Pixel, Δ: markiertes schwarzes Pixel, -: weißes Pixel als Umrandung markiert)

Die innere Konturverfolgung muss demnach nur so lange durchgeführt werden, wie der weiße Bildpunkt unterhalb der aktuellen Position nicht markiert ist.
Die Markierung der weißen Umrandung mit einer negativen Zahl wird in der Funktion „Tracer()“ durchgeführt, welche während der der Konturverfolgung aufgerufen wird.

Konturverfolgung

Ziel der Konturverfolgung ist das Auffinden der inneren und äußeren Konturen von Segmenten. Wird beim Durchlauf des Bildes ein schwarzer nicht markierter Bildpunkt gefunden, wird die Funktion „Tracer()“ aufgerufen. Handelt es sich bei dem Bildpunkt nicht um den Teil eines Segmentes sondern um einen einzelnen Punkt, so wird dies erkannt und die Konturverfolgung beendet.
Ist der Bildpunkt der Startpunkt S einer neuen Kontur, wie in Abbildung 13, dann ermittelt die Funktion „Tracer()“ den auf S folgenden Punkt der Kontur, hier mit T bezeichnet. Danach wird der auf T folgende Punkt ermittelt und so weiter. Dieser Vorgang wird so lange wiederholt, bis einerseits von der Funktion wieder der Startpunkt S zurückgeliefert wird und andererseits der auf S folgende Konturpunkt T ist. Das sind die beiden Abbruchbedingungen für die Konturverfolgung.
Die Konturverfolgung für die in Abbildung 13 dargestellte Kurve lautet: STUTSVWVS.

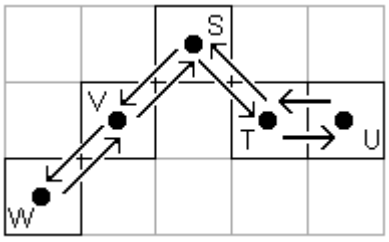


Abbildung 13: Konturverfolgung einer Kurve

Tracer()

Diese Funktion soll den für einen bekannten Konturpunkt P nachfolgenden Konturpunkt aus seiner Nachbarschaft bestimmen. Die Position jedes Nachbarpunktes von P ist wie in Abbildung 14 mit einem Index versehen.

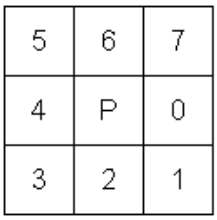


Abbildung 14: 8-Nachbarn vom Bildpunkt P

Bei der Suche des nächsten Konturpunktes werden die Nachbarn im Uhrzeigersinn durchlaufen. Der Startpunkt einer neuen Suche ist von einigen Faktoren abhängig.
Handelt es sich bei P um den Anfangspunkt einer äußeren Kontur dann wird die Startposition auf den Index 7 gelegt, da man den Punkt oberhalb des aktuellen Punktes ja bereits als weißen Bildpunkt identifiziert haben muss und folglich ist die nächste Position im Uhrzeigersinn die 7.
Handelt es sich bei P um den Startpunkt einer inneren Kontur, dann wird erst der Index für die Suche auf 3 gesetzt, da der Punkt unterhalb von P bereits als weißer Punkt identifiziert wurde.
Wurden bereits Konturpunkte gefunden und liegt der Letzte beispielsweise auf Position 3, dann wird die Suche in Richtung 5 fortgesetzt (siehe Abbildung 15), da die Position 4 bereits durchlaufen wurde. Handelt es sich bei P nicht um den Startpunkt einer inneren oder äußeren Kontur dann wird die Suchposition über

$$d + 2 (\mathrm{mod}\, 8) \qquad \textbf{(2.15)}$$

berechnet, wobei d den Index des vorigen Konturpunktes beschreibt.

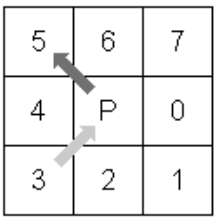


Abbildung 15: Liegt der vorige Konturpunkt auf 3 wird die Suche in Richtung 5 gestartet

Wenn die Startposition für die Suche bestimmt wurde, dann werden die Positionen im Uhrzeigersinn durchlaufen bis der erste schwarze Bildpunkt gefunden wird. Wird kein schwarzer Punkt in der Umgebung von P gefunden, so wird P als isolierter Punkt markiert. Auf der Suche nach dem nächsten schwarzen Bildpunkt, können gleichzeitig die weißen Pixel am Rand des Segments mit einer negativen Zahl markiert werden.
Wie in Abbildung 16 dargestellt, kann auf der Suche nach dem auf den Bildpunkt A folgenden Konturpunkt auch gleichzeitig der weiße Bildpunkt B am Rand des Segmentes markiert werden, bevor der nachfolgende Punkt C gefunden wird.

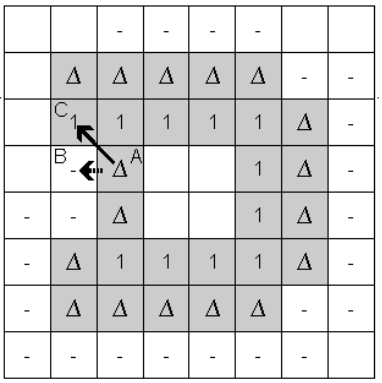


Abbildung 16: Während der Konturverfolgung Markierung der weißen Randpunkte(1: nicht markiertes schwarzes Pixel, ∆: markiertes schwarzes Pixel, -: weißes Pixel als Umrandung markiert)

Diese Verfahren funktioniert auch bei mehreren Figuren mit mehreren inneren Konturen wie in Abbildung 17 zu sehen ist. Dabei stellen die unterschiedlichen Grauwerte die einzelnen Segmente dar.

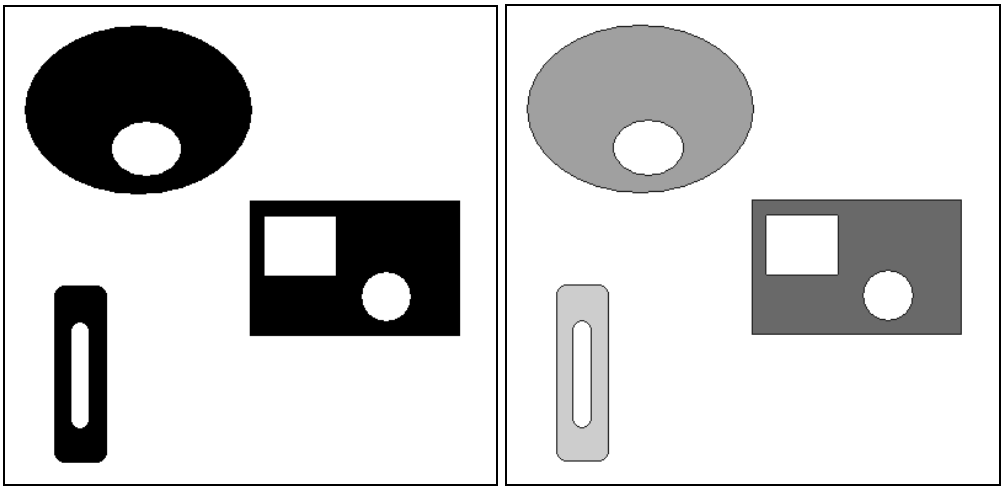


Abbildung 17: Ausgangsbild und Ergebnis der Segmentierung

Ein weiterer Vorteil der Konturverfolgung ist, dass die Randpunkte des Segments bekannt sind und in einem separaten Feld abgespeichert werden können. Durch ein Aufsummieren der Konturpunkte für jedes Segment kann dessen Umfang berechnet werden, welcher für eine Klassifizierung genutzt werden kann.

2.4. Merkmale von Segmenten

Die im letzten Abschnitt beschriebene Segmentierung liefert ein Bild, welches in Regionen eingeteilt ist. Diese Regionen können beispielsweise in einem Bild mittels unterschiedlicher Grauwerte vorliegen oder in einer Matrix mit unterschiedlichen Zahlen markiert sein. Für eine Klassifizierung werden ein oder mehrere Merkmale der Segmente benötigt.
Für die Objekterkennung werden oftmals einfache Formmerkmale herangezogen [10]. Dabei handelt es sich beispielsweise um den Flächeninhalt eines Segmentes, welcher durch das Aufsummieren aller Bildpunkte mit der gleichen Markierung berechnet wird. Ähnlich verfährt man bei der Berechnung des Umfanges einer Region, durch das Aufsummieren ihrer Randpunkte. Die Kompaktheit eines Segments wird durch das Verhältnis von Flächeninhalt zu Umfang bestimmt. Um die Lage eines Segmentes im Bild bestimmen zu können, kann man

den Schwerpunkt eines Segmentes berechnen, siehe Abbildung 19. Für die Berechnung der x-Koordinate werden alle x-Koordinaten des Segmentes aufsummiert und anschließend durch den Flächeninhalt dividiert [3]. Ebenso wird die y-Koordinate berechnet.
Für das segmentierte Bild aus Abbildung 17 werden die Merkmale der Regionen in Abbildung 18 dargestellt.

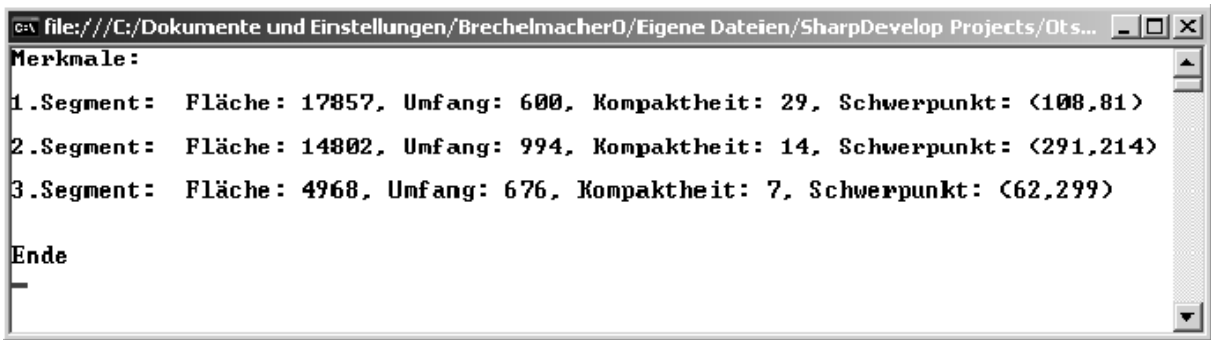


Abbildung 18: Merkmale der Segmente aus Abbildung 17

Die für die Klassifikation benötigten Merkmale eines Segments können dann in einem Merkmalsvektor zusammengefasst werden.

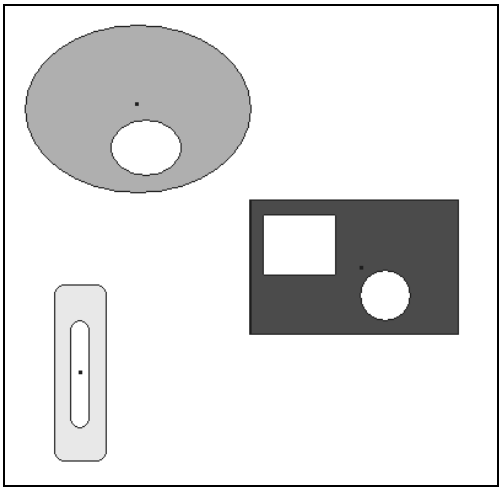


Abbildung 19: Schwerpunkte zur Positionsbestimmung der Segmente

2.5. Klassifikation

Ziel der Klassifikation ist es, Segmente aufgrund ihrer Merkmale bestimmten Klassen zuzuordnen. Die Anzahl n der berücksichtigten Merkmale spannt einen n-dimensionalen Merkmalsraum auf.

2.5.1. Der Minimum-Distance-Klassifikator

Bei diesem Verfahren wird davon ausgegangen, dass Merkmalsvektoren die nahe beieinander liegen, der gleichen Klasse angehören [10].
Für die Abstandsberechnung zwischen den Merkmalsvektoren muss das Zentrum jeder Musterklasse ermittelt werden. Dies können einerseits diejenigen Merkmalsvektoren sein, die in den Musterklassen am häufigsten auftreten, andererseits aber auch der Mittelwertvektor der Objekte der Musterklasse.

Für die Einordnung eines Objektes in eine Klasse werden zuerst die Abstände, beispielsweise der euklidische Abstand, zu den vorhandenen Zentren berechnet. Anschließend wird das Objekt der Klasse zugeordnet, zu der es den kürzesten Abstand hat. Für einen zweidimensionalen Merkmalsraum stellt die strichpunktierte Gerade in Abbildung 20 die Trennungsfunktion zwischen den beiden Musterklassen K_0 und K_1 dar. Alle Punkte auf dieser Geraden haben zu den Zentren $\underline{Z}_0$ und $\underline{Z}_1$ der beiden Klassen den selben euklidischen Abstand. Merkmalsvektoren die links der Trennungsfunktion liegen werden der Klasse K_1 zugeordnet, alle die rechts davon liegen der Klasse K_0.

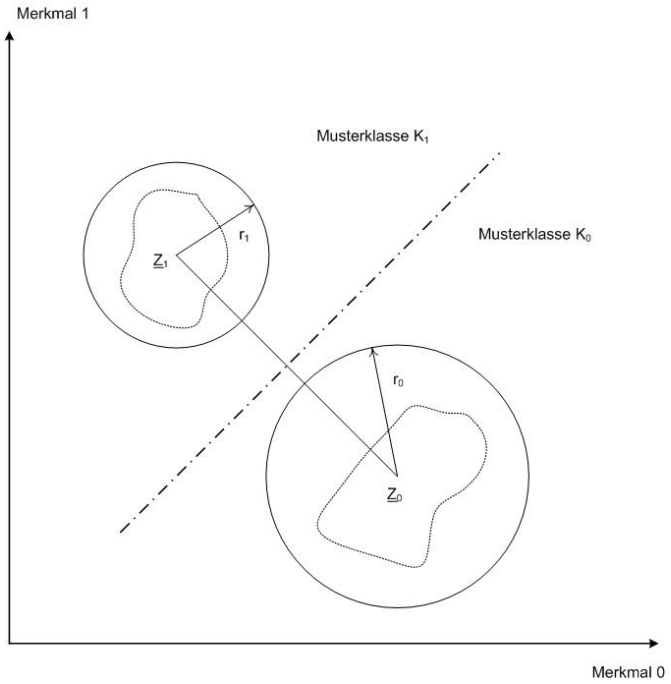


Abbildung 20: Minimum-Distance-Klassifikator

Für manche Anwendungen ist die Einführung einer Zurückweisungsklasse sinnvoll, welcher Objekte zugewiesen werden, deren Merkmale zu stark von dem Zentrum der Musterklasse abweichen. Aus diesem Grund wird jeder Klasse ein Zurückweisungsradius zugeordnet, welcher die Grenze der Musterklasse beschreibt. Ist der Abstand eines Merkmalsvektors kleiner als dieser Radius, so wird das zugehörige unklassifizierte Objekt der Klasse zugeordnet. Andernfalls wird es der Zurückweisungsklasse zugeordnet. In Abbildung 20 wird mit dem Zurückweisungsradius r_1 ein Kreis um das Zentrum $\underline{Z}_1$ aufgespannt der die Musterklasse K_1 darstellt. Alle Merkmalsvektoren die innerhalb dieses Kreises liegen werden der Musterklasse K_1 zugeordnet, liegen sie außerhalb des Kreises werden sie der Zurückweisungsklasse zugeordnet.

2.5.2. Quadermethode

Bei diesem Klassifizierungsverfahren werden die Objektklassen durch achsenparallele Quader beschrieben [10]. Alle Merkmalsvektoren die in einem Quader liegen, werden in die zugehörige Klasse eingeordnet.
Für einen zweidimensionalen Merkmalsraum ist das Prinzip in Abbildung 21 dargestellt. Alle Objekte deren Merkmalsvektoren in dem Rechteck liegen werden der entsprechenden Klasse zugeordnet. Die Musterklasse K kann dann wie folgt beschrieben werden:

$$K = \{[a_0, b_0], [a_1, b_1]\}. \quad \textbf{(2.16)}$$

Für einen n-Dimensionalen Merkmalsraum, kann die Musterklasse K_i durch folgende Zahlenpaare beschrieben werden

$$K = \{[a_0^{(i)}, b_0^{(i)}], [a_1^{(i)}, b_1^{(i)}], ..., [a_n^{(i)}, b_n^{(i)}]\}. \quad \textbf{(2.17)}$$

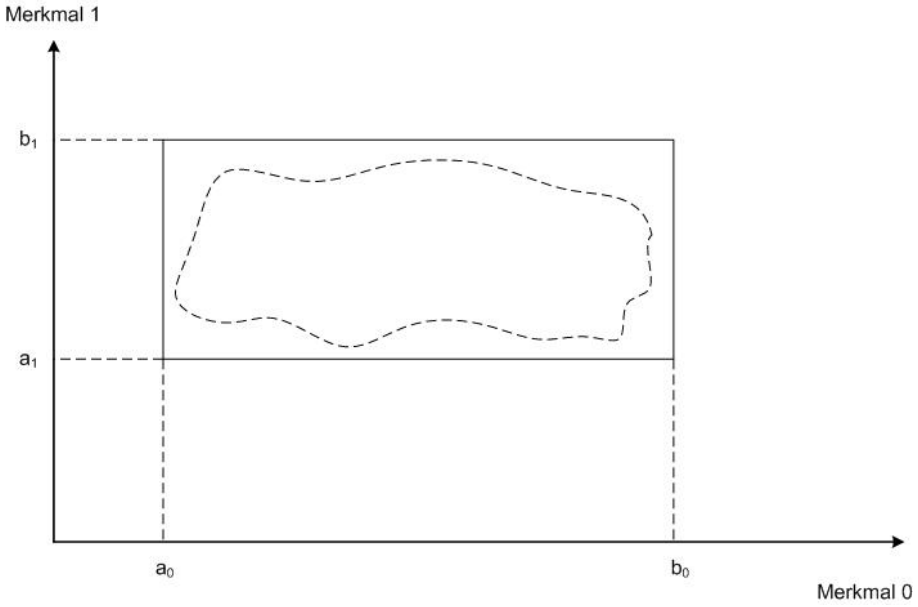


Abbildung 21: Quadermethode

Der Vorteil dieser Methode ist, dass sie sehr einfach zu implementieren ist und auch wenig Rechenzeit benötigt, da die Überprüfung ob ein Merkmalsvektor innerhalb oder außerhalb einer Klasse liegt mittels einfacher Abfragen realisiert werden kann.
Bei praktischen Anwendungen kann es vorkommen, dass sich die Quader der einzelnen Klassen überlappen. Falls ein Objekt genau in diesem Überlappungsbereich zu liegen kommt, dann kann eine Klassifizierung mit einem anderen Klassifizierungsverfahren, etwa dem Minimum-Distance-Klassifikator, durchgeführt werden.

2.6. Bildregistrierung

Das Ziel der Bildregistrierung ist es nach [12], einen Zusammenhang zwischen zwei unterschiedlichen Bildern herzustellen, welche dieselbe Szene darstellen. Dadurch können Bilder eines oder mehrerer Objekte

- die zu unterschiedlichen Zeiten
- aus unterschiedlichen Perspektiven
- mit verschiedenen Verfahren

aufgenommen wurden miteinander verglichen werden. Mittels des Registrierungsverfahrens werden korrespondierende Objekte einander zugeordnet.
Dies geschieht nach [5] in drei Schritten:

1. Erfolgt die Auswahl von markanten Merkmalen, beispielsweise Kanten oder Blobs. Wichtig dabei ist, dass diese Merkmale reproduzierbar sind und auch unter anderen Aufnahmebedingungen gefunden werden.
2. Die Nachbarschaft der ausgewählten Punkte wird durch Merkmalsvektoren beschrieben, welche unabhängig von Rauschen, Fehler in der Detektion sowie von geometrischen Verzerrungen und Verzerrungen bei der Bildaufnahme sein müssen.

3. Zuletzt müssen die Merkmalsvektoren der beiden Bilder miteinander verglichen und einander zugeordnet werden. Dies kann beispielsweise über den euklidischen Abstand der Vektoren zueinander erfolgen.

Die Dimension der Merkmalsvektoren hat einen großen Einfluss auf die Rechenzeit, deshalb sind Vektoren mit möglichst wenigen Merkmalen wünschenswert. Jedoch kann die Dimension der Merkmalsvektoren nicht beliebig reduziert werden, da es sonst unmöglich ist eine Transformation zu finden welche die Bilder einander angleicht. Für den Fall einer linearen Verschiebung zweier Bilder zueinander, reduziert sich die Transformation auf einen Verschiebungsvektor. Um dies zu verdeutlichen wurde in Abbildung 22 zwei Bilder von denselben Objekten aufgenommen, wobei die Kameraposition leicht verändert wurde. Die Objekte des linken Teilbildes sind gegenüber den Objekten des rechten Teilbildes leicht verschoben. Der Merkmalsvektor für die Zuordnung der Objekte könnte aus dem Grauwert und der Objektgröße bestehen. Ebenso können die Abstände zwischen den Objekten im linken und den Objekten im rechten Teilbild berechnet werden und eine Zuordnung der Objekte über den jeweils kürzesten Abstand erfolgen.

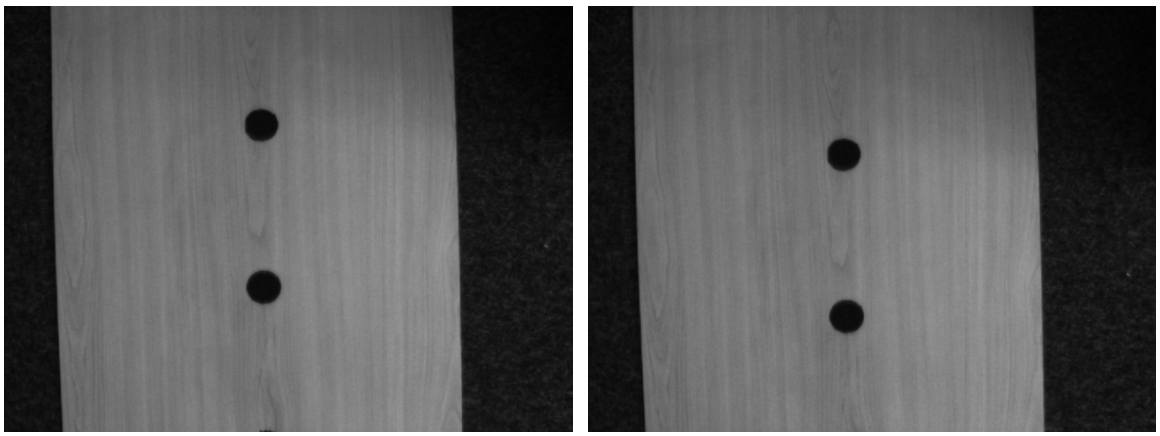


Abbildung 22: Gleiche Objekte bei Verschiebung der Kamera

3. Das Demonstratorsystem

Für den Aufbau des Demonstratorsystems ist die Konstruktion eines verschiebbaren Versuchsaufbaus notwendig, welcher sämtliche Komponenten aufnehmen kann. Ebenso ist die Planung und Fertigung der Ausleger zur Aufnahme der Filzstifte erforderlich.

3.1. Der Versuchsaufbau

Die Eckdaten für den Versuchsaufbau wurden wie folgt vorgegeben:

- Der Abstand von der Kamera zum Boden soll 110cm betragen.
- Der Abstand zwischen den Rädern (quer zur Fahrtrichtung) soll 60cm betragen.
- Der Abstand von den Motoren zur Kamera soll ungefähr 80cm betragen, am Besten jedoch wäre es, wenn die Kameraposition verändert werden könnte.

Das Demonstratorsystem besteht weiters aus einem Rechner in einem 19 Zoll Industriegehäuse, einem Flachbildschirm, einer Farbkamera von The Imaging Source, zwei Servomotoren mit den zugehörigen Frequenzumrichtern sowie Ethernetoperatoren und EMV-Filter von KEB und einem D-Link Switch. Um den Versuchsaufbau möglichst einfach verschieben zu können, werden alle Komponenten darauf montiert, damit nur eine Kabelrolle für die Stromversorgung des gesamten Demonstratorsystems benötigt wird.

Abbildung 23: Skizze des Aufbaus

Da neben den oben angeführten Komponenten noch für jede Frequenzumrichter-Servomotor-Einheit jeweils 20 Meter Leistungskabel und Resolverkabel vorhanden sind, wurde der Versuchsaufbau wie in Abbildung 23 geplant. Dabei handelt es sich um einen Wagen, der auf vier Rollen geschoben werden kann. An der Front befindet sich ein Ausleger mit einer verschiebbaren Kamerahalterung (Abbildung 24 linkes Teilbild). Diese Halterung wurde so konstruiert, dass die Kamera trotz Montageplatte in der Mitte montiert wird. Da jedoch nach

der Fertigung der Halterung ein anderes Kameramodell benutzt wurde, befindet sie sich nicht mehr exakt zwischen den Motoren.
Das rechte Teilbild aus Abbildung 24 zeigt die Platte zur Aufnahme der beiden Servomotoren, welche unterschiedliche Gehäuse und Wellendurchmesser besitzen. Die notwendigen Abmessungen für die Montage der Motoren wurden dem Datenblatt [8] entnommen.

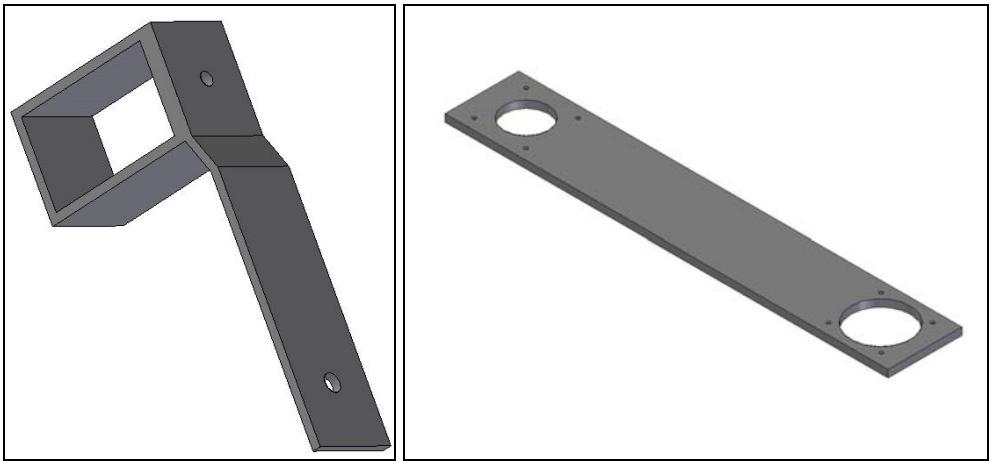


Abbildung 24: Kamerahalterung(links), Motorplatte(rechts)

Die untere Platte dient zur Aufnahme der Motor- und Geberleitungen. Ebenso wurde eine schaltbare Steckerleiste für die Stromversorgung aller Komponenten und der D-Link Switch montiert. Auf der oberen Platte wurden der Flachbildschirm, der Industrie-PC sowie die EMV-Filter und die Frequenzumrichter montiert. Letzte werden auf die Filter geschraubt und bilden so eine Einheit. Die Ethernet Operatoren werden auf die Frequenzumrichter gesteckt und mit dem Switch verkabelt. Abbildung 25 zeigt den fertigen Aufbau des Demonstratorsystems.

Abbildung 25: Fertiger Versuchsaufbau des Demonstratorsystems

Für die Montage der Filzstifte an die Motorachsen wurden Ausleger konstruiert. Das linke Teilbild in Abbildung 26 zeigt den ersten Ausleger. Dieser ist 25cm lang und aus Eisen gefertigt. Doch für die Positionierung war dieser aufgrund des zu großen Massenträgheitsverhältnisses nicht geeignet. Selbst nachdem 23 Löcher mit 10mm Durchmesser zur Massenreduktion gebohrt wurden, konnte der Positionsregler des Frequenzumrichters den Ausleger nicht exakt positionieren und begann um die Zielposition zu schwingen. Aus diesem Grund wurde der im linken Teilbild abgebildete Ausleger aus PVC gefertigt. Da der Werkstoff deutlich leichter ist und einfacher bearbeitet werden kann wurde im Gegensatz zum Ausleger aus Metall ein Z-Profil realisiert, wodurch der Filzstift näher am Boden geführt werden kann. Die Befestigung des Filzstiftes erfolgt durch eine Klemmung. Für ein besseres Massenträgheitsverhältnis wurde der Ausleger noch zusätzlich mit Bohrungen versehen.

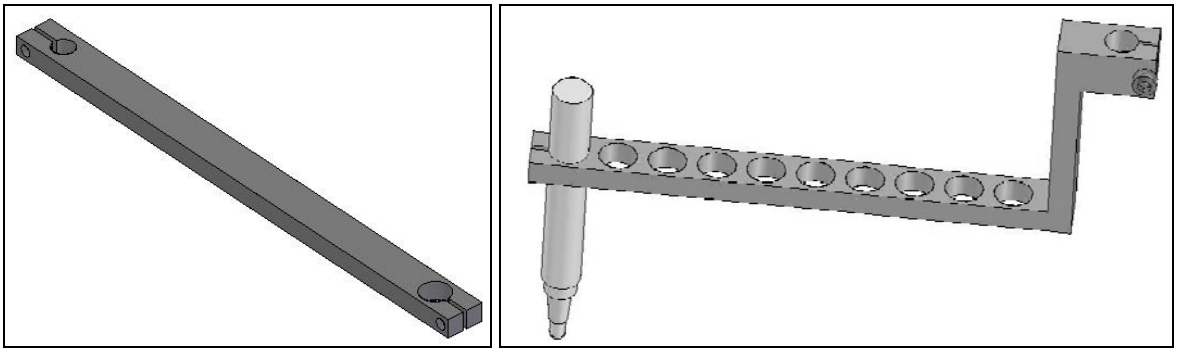


Abbildung 26: Ausleger aus Eisen (links), Ausleger aus PVC (rechts)

Die Pläne für die Fertigung entstanden mit Solid Edge V16.

3.2. Die Hardware

Die Hardware des Demonstratorsystems setzt sich aus mehreren Komponenten zusammen. Für die Bildaufnahme wird eine The Imaging Source Farbkamera mit einer Auflösung von 640x480 Bildpunkten verwendet. Die Anbindung der Kamera an den Rechner erfolgt über die Schnittstelle FireWire. Vom Kamerahersteller wird Software mitgeliefert, welche eine Steuerung der Kamera mit C# ermöglicht.
Der Rechner wird für die Steuerung des Demonstratorsystems benötigt. Da die einzelnen Schritte der Bildverarbeitung sehr rechenintensiv sind wurde ein Rechner mit 4 Prozessorkernen verwendet.
Bei den Frequenzumrichtern werden Frequenzumrichter der Baureihe F5 von der Firma KEB verwendet. Dabei handelt es sich um einen geregelten, feldorientierten Frequenzumrichter für Drehstrommotoren, welcher mit einer Phase betrieben wird [6].
Zur Einhaltung der EMV-Richtlinien wird ein Filter zwischen den Frequenzumrichter und das Stromnetz geschalten. Der EMV-Filter ist ebenfalls von KEB und speziell für die Baureihe F5 konzipiert, weshalb der Frequenzumrichter auf diesen geschraubt und mit den vorhandenen Kabeln verbunden wird. Die modulare Bauweise der einzelnen Komponenten ermöglicht eine kompakte Montage der Einheit auf dem Versuchsaufbau. Über eine Diagnoseschnittstelle ist das Schreiben und Lesen von Parameterwerten beziehungsweise von Parameterlisten mittels der Software des Herstellers möglich. Um die im Frequenzumrichter gespeicherten Parameterlisten abzuarbeiten, muss die Reglerfreigabe betätigt werden. Dies ist auch für das Löschen eines Fehlers notwendig. Deshalb wurde eine Halterung gefertigt und an den Griff

des Versuchaufbaus montiert, sodass die Reglerfreigabe auch während des Betriebs betätigt werden kann.
Für eine lokale oder externe Bedienung des Frequenzumrichters wird ein Operator benötigt. Dieser verfügt neben einem kleinen Bedienfeld für die lokale Bedienung auch über eine, für die Kommunikation über das Bussystem notwendige, Schnittstelle. Diese Operatoren werden für unterschiedliche Bussysteme angeboten. Für die Kommunikation der Frequenzumrichter mit dem Rechner stehen Ethernet-Operatoren von KEB zur Verfügung. Diese verfügen über eine RJ45-LAN-Schnittstelle und eine Diagnoseschnittstelle, welche zum Auslesen und Schreiben von Parameterwerten mit der Combivis5 Software genutzt werden kann [7]. Der Operator wird auf die Oberseite des Frequenzumrichters gesteckt und bietet, neben der Steuerung über eine Tastatur, auch die Anbindung an ein Netzwerk. Die IP-Adresse kann für jeden Ethernet-Operator direkt am Gerät eingestellt werden. Ebenso ist es möglich durch die Eingabe der IP-Adresse in einem Webbrowser die Umrichterparameter auszulesen. Bei den Servomotoren handelt es sich um Drehstromsynchronmotor aus der 200V-Klasse [8]. Die Servomotoren verfügen über zwei drehbare Winkelflanschdosen, einmal den 8-poligen Servomotor Leitungsstecker und den 12-poligen Resolveranschluss. Über diese beiden Leitungen ist der Servomotor mit dem Frequenzumrichter verbunden.
Damit eine Kommunikation mit beiden Frequenzumrichtern möglich ist werden diese über eine D-Link Switch mit dem Rechner verbunden.

4. Motorsteuerung

Die Kommunikation zwischen dem Rechner und den Servomotoren erfolgt über Ethernet. Deshalb sind die Frequenzumrichter mit Ethernet-Operatoren ausgestattet, welche über eine IP-Adresse verfügen und somit direkt angesprochen werden können.
Dies ist einerseits mittels der Software Combivis5 des Herstellers möglich oder über Telegramme des erweiterten DIN66019II Protokolls.

4.1. Steuerung mittels Combivis5

KEB stellt für das Arbeiten mit Frequenzumrichtern das Programm Combivis5 zur Verfügung. Beim Starten des Programms wird versucht zu einem oder mehreren Umrichtern eine Verbindung aufzubauen, was einerseits über die Diagnoseschnittstelle andererseits über das Netzwerk möglich ist. Kann eine Verbindung hergestellt werden, dann bietet die Software die Möglichkeit, Parameter des Umrichters auszulesen oder zu verändern. In Abbildung 27 ist die Arbeitsoberfläche zu sehen. Auf der linken Seite befinden sich für den an Knoten 0 angeschlossenen Frequenzumrichter die Umrichterparameter. Diese sind in Gruppen zusammengefasst, wie etwa die Betriebsdaten-Anzeige, welche einen Überblick über die aktuelle Position oder etwa die Geschwindigkeit liefert, oder die „Posi-/Synchronparameter“, wo die für die Positionierung oder den Synchronlauf notwendigen Parameter eingestellt werden können. Auf der rechten Seite befinden sich dann alle Parameter der ausgewählten Gruppe.

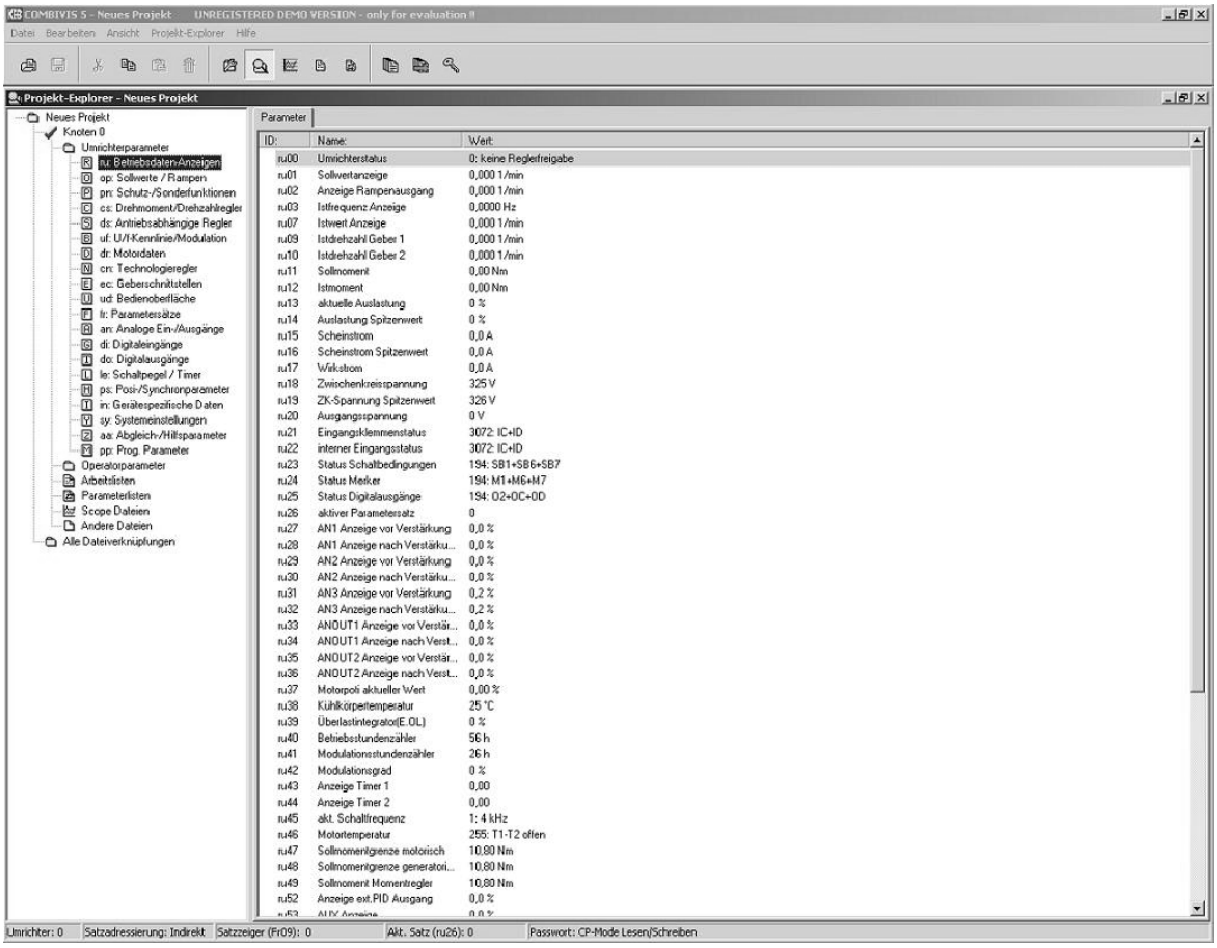


Abbildung 27: Combivis5 Software

Die Parameter für die Positionierung beziehungsweise den Synchronlauf sind in Abbildung 28 zu finden. Es besteht im Positionierbetrieb die Möglichkeit bis zu 16 Positionen zu speichern und anzufahren.

ID:	Name:	Wert:
▶ PS00	Posi-/Synchronmodus	165: Posimodus+aus+ps.25/ps.25+Letzte Ziehlposition+neuer Versuch+aus+aus+ein+aus
▶ PS01	Master Quelle	0: Kanal1
PS02	Posi/Synch Eingangswahl	1: ST
PS03	Slavekorrekt.Eingangsw.	0: kein Eingang
PS04	Slavekorrektur	0 Inc
▶ PS05	Startoffset	0 Inc
▶ PS06	KP für Posi/Synchron	500
▶ PS09	Lagegrenze Posi/Synchron	500,000 1/min
PS10	Slavekorrekt.inv.Eingangswahl	0: kein Eingang
PS11	Rst MA/Sl.diff.Eingangswahl	0: kein Eingang
PS13	Ref.punkt setzen Eingangwahl	0: kein Eingang
PS14	Referenzpunktfahrt Modus	0: aus+nein+rechts+aus+aus+aus+nein+aus+aus
PS15	Software-Endschalter links	-1073741824 Inc
PS16	Software-Endschalter rechts	1073741823 Inc
PS17	Referenzpunkt	0 Inc
PS18	Referenzpunkt Eingangswahl	0: kein Eingang
PS19	Start Refpkt. Eingangswahl	0: kein Eingang
PS20	Refpktfahrt Rampenzeit	0,00 s
PS21	Refpktfahrt Geschwindigkeit	20,000 1/min
PS23	Index/Auswahl	0
PS24	Index/Position	0 Inc
PS25	Index/Geschwindigkeit	0,000 1/min
PS26	Index/nächster	0
PS27	Index/Modus	1: ja+absolut
▶ PS28	Startindex neues Profil	0
PS29	Start Posi Eingangswahl	1: ST
PS30	Zielfenster	1024 Inc
PS31	max. Geschwindigkeit %	100,0 %
PS33	Bahnmodus Sollwertquelle	0: PS.34
PS34	Bahnmodus Sollwert	0 Inc
PS35	Teachmodus	0: schreibe Index PS.23
PS36	Teach Index Eingangswahl	0: kein Eingang
PS37	Lageabtastung Eingangswahl	0: kein Eingang
PS52	Autom.Positionierung nach S...	0: aus

Abbildung 28: Parameter für Positionierung

Es gibt eine Vielzahl von Einstellmöglichkeiten und in Abbildung 29 ist das Fenster für die Einstellungen des Parameters „PS00 Posi-/Synchronmodus“. Hier kann die Auswahl zwischen Positionierung und Synchronlauf getroffen werden. Ebenso kann das Verhalten im Fehlerfall und auch für den Fall, dass die Position nicht erreichbar ist, festgelegt werden. Der Wert für den Parameter „PS00“, welcher auf den Frequenzumrichter geschrieben wird, setzt sich aus den Werten der einzelnen Punkte zusammen und ergibt in diesem Fall 165 als Summe von 5+32+128.

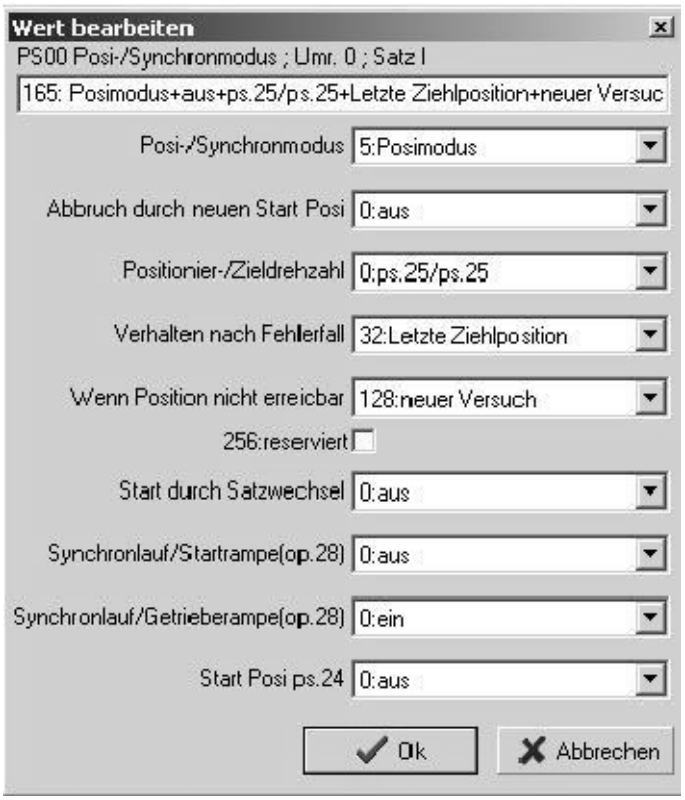


Abbildung 29: Fenster für „Posi-/Synchronmodus“

Ebenso wurde der Parameter „PS27“ siehe Abbildung 30 auf den Wert 1 gesetzt. Das Fortsetzen der Profilabarbeitung bedeutet, dass nach dem Erreichen einer Position automatisch auf die nächste gespeicherte Position gewechselt wird. Die Positionsvorgabe erfolgt absolut, da die auf dem Motor montierten Ausleger ausgerichtet wurden.

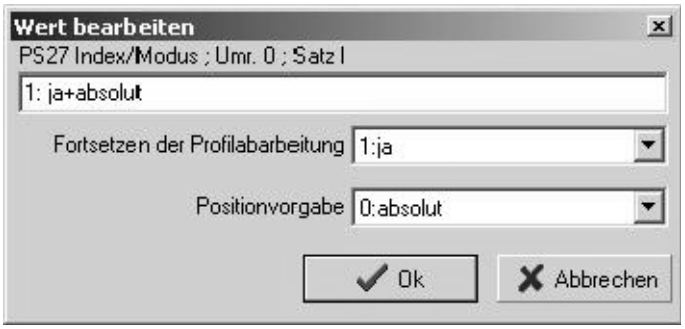
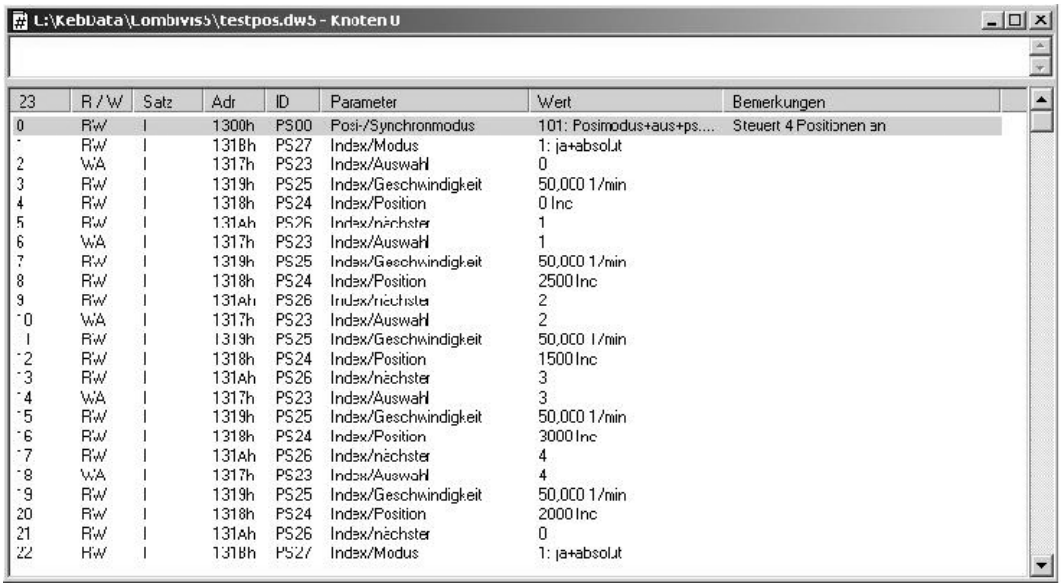


Abbildung 30: Fenster für „Index/Modus“

Mittels Parameterlisten können Programme auf den Frequenzumrichter geschrieben werden. Dazu können die Parameter von der Parametergruppe des Hauptfensters in eine Parameterliste gezogen oder eingefügt und anschließend verändert werden. Hat das Programm die notwendige Funktionalität kann es an den Frequenzumrichter geschickt werden. Nach Betätigung der Reglerfreigabe wird die neue Parameterliste ausgeführt. Abbildung 31 zeigt eine Parameterliste mittels welcher der „Posimodus“ aktiviert wird und die 5 Positionen 0, 2500, 1500, 3000 und 2000 angesteuert werden. Die Geschwindigkeit wurde für jeden Index auf 50 Umdrehungen pro Minute gesetzt und jeweils der nachfolgende Index angegeben.
Es auch möglich jeden Index mit einer unterschiedlichen Geschwindigkeit anzufahren oder von Index 1 auf Index 4 zu wechseln.



23	R / W	Satz	Adr	ID	Parameter	Wert	Bemerkungen
0	RW	I	1300h	PS00	Posi-/Synchronmodus	101: Posimodus+aus+ps....	Steuert 4 Positionen an
1	RW	I	1318h	PS27	Index/Modus	1: ja+absolut	
2	WA	I	1317h	PS23	Index/Auswahl	0	
3	RW	I	1319h	PS25	Index/Geschwindigkeit	50,000 1/min	
4	RW	I	1318h	PS24	Index/Position	0 Inc	
5	RW	I	131Ah	PS26	Index/nächster	1	
6	WA	I	1317h	PS23	Index/Auswahl	1	
7	RW	I	1319h	PS25	Index/Geschwindigkeit	50,000 1/min	
8	RW	I	1318h	PS24	Index/Position	2500 Inc	
9	RW	I	131Ah	PS26	Index/nächster	2	
10	WA	I	1317h	PS23	Index/Auswahl	2	
11	RW	I	1319h	PS25	Index/Geschwindigkeit	50,000 1/min	
12	RW	I	1318h	PS24	Index/Position	1500 Inc	
13	RW	I	131Ah	PS26	Index/nächster	3	
14	WA	I	1317h	PS23	Index/Auswahl	3	
15	RW	I	1319h	PS25	Index/Geschwindigkeit	50,000 1/min	
16	RW	I	1318h	PS24	Index/Position	3000 Inc	
17	RW	I	131Ah	PS26	Index/nächster	4	
18	WA	I	1317h	PS23	Index/Auswahl	4	
19	RW	I	1319h	PS25	Index/Geschwindigkeit	50,000 1/min	
20	RW	I	1318h	PS24	Index/Position	2000 Inc	
21	RW	I	131Ah	PS26	Index/nächster	0	
22	RW	I	1318h	PS27	Index/Modus	1: ja+absolut	

Abbildung 31: Parameterliste (5 Positionen werden angefahren)

Die Combivis5 Software beinhaltet auch ein „Scope“. Dabei handelt es sich um eine Funktion mit der man Umrichterparameter wie auf einem Oszilloskop darstellen kann.
In Abbildung 32 werden die Istposition (ru54), die Sollposition (ru56) und der aktuelle Positionsindex (ru60) dargestellt. Die Istposition wird durch die mittlere Kurve und die Sollposition durch die obere Kurve beschrieben. Die beiden Parameter weichen nur geringfügig voneinander ab, da einerseits keine Last an der Motorachse befestigt wurde und andererseits die Geschwindigkeit des Positioniervorganges relativ niedrig gewählt wurde. Die unterste Kurve beschreibt den verwendeten Positionsindex, welcher erst nach dem Erreichen der Position gewechselt wird.

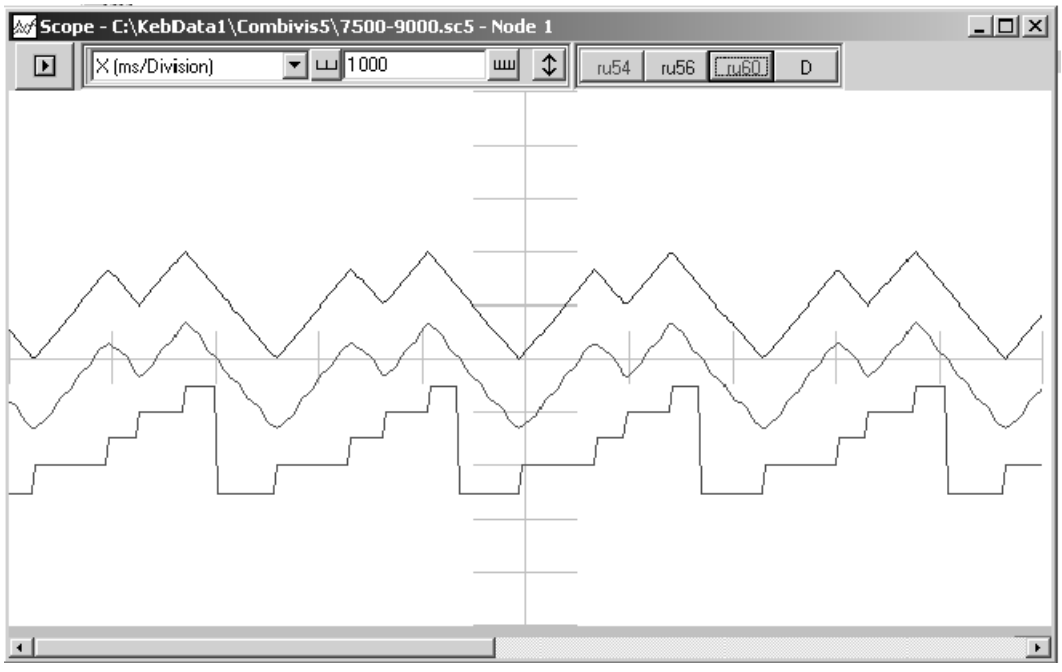


Abbildung 32: „Scope“ der Parameterliste aus Abbildung 31

4.2. Motorsteuerung mit C#

Die Software zur Steuerung der Motoren wurde in C# mittels Visual Studios implementiert. Sie beinhaltet einigen Grundfunktionen, wie etwa den Verbindungsaufbau oder das Umwandeln von Zahlen von dezimal auf hexadezimal, sowie darauf aufbauende Funktionen wie etwa die Initialisierung eines Frequenzumrichters oder die Änderung der Positioniergeschwindigkeit. Für die Kommunikation mit den Frequenzumrichtern über Ethernet werden Telegramme des DIN66019II Protokolls benötigt.
Für den Positionierbetrieb wurden zwei Modi implementiert. Der erste Modus verwendet nur einen Index zur Positionierung. Das heißt, dass der Motor auf eine Position gestellt wird und so lange dort verharrt, bis ihm eine neue Position übermittelt wird. Der zweite Modus benutzt zwei Indizes für die Positionierung, so dass immer zwischen den beiden Positionen hin- und hergefahren wird, was eine schwingende Bewegung der Ausleger erzeugt. Hier können beide Positionen verändert werden, aber auch eine Position fixiert und die andere verändert werden.

4.2.1. DIN66019II Protokoll

Das DIN66019II Protokoll ermöglicht mittels Diensten das Lesen und Schreiben von Parametern, Parameterkonfigurationen sowie Prozessdaten auf beziehungsweise von den Frequenzumrichtern. Sämtliche Dienste und der genaue Aufbau der Telegramme sind in der Dokumentation des Herstellers [9] zu finden.

Steuerzeichen

Die für die Kommunikation notwendigen Steuerzeichen sind in Tabelle 1 zu finden.

EOT = 04h	ENQ = 05h	STX = 02h	ETX = 03h	ACK = 06h	NAK = 15h

Tabelle 1: Steuerzeichen

Da die Steuerzeichen in einem Bereich unter 20h liegen, können sie von den ASCII-Codierten Zeichen unterschieden werden und werden deshalb uncodiert gesendet.

BCC

Ein wichtiger Parameter ist der „Block Check Charakter BCC“. Er dient zur Überprüfung der Vollständigkeit beziehungsweise Richtigkeit der übermittelten Daten. Um den „Block Check Charakter“ zu berechnen, erfolgt eine Exklusiv-Oder Verknüpfung der Daten. Dieser Wert muss größer als 20h sein, um ihn von den Steuerzeichen unterscheiden zu können. Ist er das nicht, wird er um 20h erhöht. Der „Block Check Charakter“ befindet sich am Ende jedes Telegramms, welches an den Frequenzumrichter geschickt wird.

IID

Bei der „Invoke-ID IID“ handelt es sich um eine hexadezimale Zahl im Bereich von 1 bis F. Sie wird dazu benutzt, um die Response des Umrichters der entsprechenden Request zuordnen zu können. Nach jeder Request wir die „Invoke-ID“ erhöht, so dass sich zwei aufeinander folgende Requests um den Wert 1 unterscheiden.

Dienst

Der jeweilige Dienst wird als 1 Zeichen in ASCII-Codierung mit einem Offset von 47h übertragen. So wird der Dienst 0 als 47h dargestellt, was dem Zeichen G entspricht.

Negative Quittungen

Tritt während der Kommunikation ein Fehler auf, so wird eine negative Quittung zurückgeschickt. Sie ist in Abbildung 35 dargestellt und besteht aus drei Teilen. Der erste Teil ist die „Invoke-ID“ welche angibt, auf welchen Request sie sich bezieht. Der zweite Teil beschreibt den aufgetretenen Fehler mittels eines hexadezimalen Zeichens. Zuletzt wird das Steuerzeichen „NAK“ übertragen.
In der Software wurde die Fehlerbehandlung so implementiert, dass beim Auftreten einer negativen Quittung, welche an dem Steuerzeichen erkannt wird, eine Exception an die nächst höhere Programmebene weitergegeben wird, die angibt bei welchem Befehl ein Fehler aufgetreten ist. Für die Anwendung ist es nicht wichtig welcher Fehler aufgetreten ist, da keine explizite Fehlerbehandlung erfolgt. Es wird eine Fehlermeldung generiert, die dem Benutzer mitteilt an welcher Stelle im Programm der Fehler aufgetreten ist.

Schreiben eines Parameterwertes

Der Dienst 0/1 des DIN66019II Protokolls dient zum Schreiben von Parameterwerten. Dazu muss ein Datenrahmen wie in Abbildung 33 erstellt werden.

STX	Dienst	IID	Parameteradresse ASCII				Parameterdaten ASCII								Satz ASCII		ETX	BCC
02h	H	2	2	1	0	1	F	F	F	F	A	2	4	0	0	1	03h	>1Fh

Abbildung 33: Schreiben eines Parameterwertes

An erster Stelle steht das Steuerzeichen „STX“. Im Anschluss daran befindet sich die Nummer des Dienstes, mit einem Offset von 47h, gefolgt von der „Invoke-ID“. Die nächsten vier Zeichen sind die Adresse des zu schreibenden Parameters, gefolgt von acht Zeichen, welche den Wert des Parameters darstellen. Die Adressen der Parameter sind in der Parameterliste zu finden (siehe Abbildung 31), stehen jedoch nicht in den einzelnen Parametergruppen des Hauptfensters. Die folgenden 2 Zeichen geben die Nummer des benutzten Satzes an, gefolgt von dem Steuerzeichen „ETX“ und dem „Block Check Charakter“. Für die Parameterdaten und den „Block Check Charakter“ wurden eigene Funktionen implementiert, welche später beschrieben wird.
Wird der Datenrahmen fehlerfrei empfangen und liegt der Parameterwert innerhalb der vorgegebenen Grenzen, so wird eine Response (wie in Abbildung 34), von dem Frequenzumrichter zurückgeschickt.

IID	ACK
2	06h

Abbildung 34: Empfangsbestätigung

Werden mehrere Datenrahmen geschickt, so gibt die „Invoke-ID“ an, welcher fehlerfrei empfangen wurde oder wo Probleme aufgetreten sind. In diesem Fall wird eine negative Quittung zurückgeschickt, welche die Nummer des fehlerhaften Request und die Art des Fehlers beinhalten.

IID	Fehler	NAK
2	2	15h

Abbildung 35: Empfangsfehler

Abgeschlossen wird die negative Quittung mit dem Steuerzeichen „NAK“. Die Auswertung des Steuerzeichens am Ende eines empfangenen Paketes kann deshalb für eine Fehlerbehandlung genutzt werden.

Lesen eines Parameterwertes

Das Lesen eines Parameterwertes erfolgt mit dem Dienst 0/1 des DIN66019II Protokolls. Das dafür notwenige Telegramm ist in Abbildung 36 dargestellt.

Dienst	IID	Parameteradresse ASCII				Satz ASCII		ENQ	BCC
G/H	1	2	0	0	0	0	1	05h	>1Fh

Abbildung 36: Request für das Lesen eines Parameters

An erster Stelle befindet sich die Bezeichnung des Dienstes, gefolgt von der „Invoke-ID“. Die folgenden vier Stellen geben die Adresse des Parameters an, dessen Wert gelesen werden soll. Anschließend wird die Satzkennung angegeben, gefolgt von dem Steuerzeichen „ENQ“ und dem „Block Check Charakter“.
Die Response auf diese Anfrage wird in Abbildung 37 dargestellt.

STX	Dienst	IID	Parameterdaten ASCII								ETX	BCC
02h	G/H	1	0	0	0	0	0	0	4	6	03h	>1Fh

Abbildung 37: Telegramm mit dem Parameterwert

Dieses Telegramm enthält am Anfang das Steuerzeichen „STX“ und am Ende „ETX“ sowie den „Block Check Charakter“. Die „Invoke-ID“ wird dazu genutzt um den achtstelligen Parameterwert einer Anfrage und somit einem Parameter zuzuordnen.
Für den Fall, dass ein Fehler aufgetreten ist, wird ein Telegramm wie in Abbildung 38 zurückgeschickt.

IID	Fehler	EOT
1	3	04h

Abbildung 38: Fehler beim Lesen eines Parameters

Im Gegensatz zur negativen Quittung beim Schreiben von Daten enthält das Telegramm neben der „Invoke-ID“ und dem Fehlercode am Schluss das Steuerzeichen „EOT“.

4.2.2. Grundfunktionen

Für den Betrieb der Frequenzumrichter sind einige Funktionen notwendig, die von mehreren Unterprogrammen genutzt werden. Dazu zählt etwa die Berechnung des „Block Check Charakters“ aber auch der Aufbau oder das Trennen der Verbindung.

Fehlerbehandlung

Treten während des Programmablaufs Fehler auf, etwa durch den Abbruch der Netzwerkverbindung oder den Empfang einer negativen Quittung, so kann das aktuelle Programm nicht fortgesetzt werden, da es beispielsweise keinen Sinn macht weiterhin Telegramme an einen Frequenzumrichter zu schicken, der nicht erreichbar ist.
C# bietet zur Behandlung solcher Probleme Ausnahmeobjekte aus der Klasse System.Exception an [1]. Einerseits besteht die Möglichkeit vorhandene Exceptions zu verwenden, wie sie etwa beim Scheitern des Verbindungsaufbaus vom System erzeugt

werden, oder selbst Fehlermeldungen in Form von Exceptions zu generieren, um den Empfang einer negativen Quittung zu signalisieren.
Da es nicht immer sinnvoll ist den Fehler gleich zu bearbeiten oder eine Fehlermeldung auszugeben, können Exceptions an die nächst höhere Programmebene, welche das Unterprogramm aufgerufen hat, weitergereicht werden.
In Abbildung 39 wird die in der Funktion „Verbinden()“ auftretende SocketException bis in das Hauptprogramm weitergereicht. Dies hat den Vorteil, dass die Fehlermeldung an die jeweilige Situation angepasst werden kann. Wenn die Funktion „Verbinden()“ von mehreren Unterprogrammen genutzt wird, so kann ein Verbindungsfehler nicht zugeordnet werden. Es ist jedoch möglich die Fehlermeldung bei jedem Weiterreichen zu modifizieren, sodass beispielsweise der Name des Unterprogramms eingefügt wird.

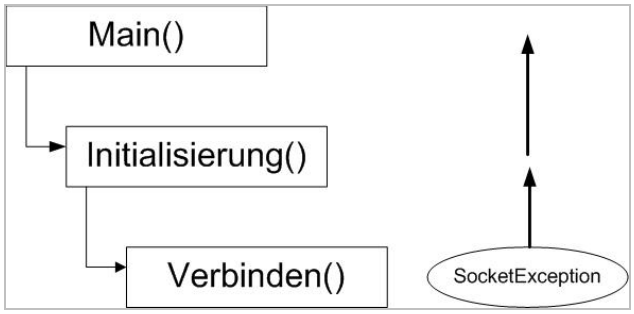


Abbildung 39: Exception wird an das Hauptprogramm weitergereicht

Dadurch bietet sich die Möglichkeit unterschiedlich auf Fehler zu reagieren. Kann keine Verbindung aufgebaut werden so kann im Unterprogramm „Initialisierung()“ die Funktion „Verbindung()“ nochmals aufgerufen werden. Dieser Vorgang kann so lange wiederholt werden bis es klappt, oder eine bestimmte Anzahl an Verbindungsversuchen gescheitert ist und erst dann wird eine Ausnahme generiert, um den Hauptprogramm mitzuteilen, dass eine Initialisierung nicht möglich ist.
Abbildung 40 zeigt wie eine Ausnahme abgefangen wird. Im try-Block steht der abzuarbeitende Programmteil. Tritt in diesem Programmteil eine Exception auf, die der Ausnahme-Klasse der catch Abfrage entspricht, so erfolgt die Abarbeitung des Programms im catch-Block. Dort kann beispielsweise eine Fehlermeldung ausgegeben oder an eine höhere Programmebene weitergereicht werden. Es ist auch möglich mehrere catch-Blöcke zu benutzen um auf unterschiedliche Ausnahmen reagieren zu können. Ebenso ist es auch möglich die catch-Abfrage ohne Angabe einer Ausnahme-Klasse zu verwenden, dann werden dort sämtliche Exceptions behandelt.

```
try
{
        Dieser Block enthält das zu überwachende Programm
}
catch (Ausnahme-Klasse Parametername)
{
        Behandlung der Ausnahme
}
```

Abbildung 40: Aufbau einer Ausnahmebehandlung

Verbinden()

Die Funktion „Verbinden()“ dient zum Aufbau einer Verbindung mit einem Frequenzumrichter. Sie bekommt dessen IP-Adresse als String übergeben und überprüft, ob bereits eine Verbindung zu dem gewünschten Frequenzumrichter vorhanden ist. Ist dies nicht der Fall, wird eine Socket-Verbindung aufgebaut. Für die Kommunikation mit den Frequenzumrichtern wird der Protokolltyp auf TCP gesetzt und der Port mit dem Wert 8000 initialisiert. Für einen Verbindungsaufbau wird die IP-Adresse des Teilnehmers und die Portnummer des Dienstes benötigt um eine virtuelle Verbindung zwischen zwei Anwendungen aufbauen zu können. Auf beiden Seiten stehen Programmierschnittstellen zur Verfügung, die als Sockets bezeichnet werden. Diese kommunizieren über Datenströme, in Form von Bytes, miteinander [1]. Hierbei handelt es sich um die Sitzungsebene des OSI-Modells, welche für die Kommunikation zwischen zwei Anwendungen zuständig ist.
Für den Fall, dass keine Verbindung aufgebaut werden kann, wird eine SocketException erzeugt und an die übergeordnete Programmebene geschickt.

Trennen()

Diese Funktion dient zum Trennen der Verbindung mit einem Frequenzumrichter. Dazu wird überprüft, ob eine Socket-Verbindung vorhanden ist. Falls ja, wird diese getrennt. Die Angabe der IP-Adresse ist für das Trennen der Verbindung nicht erforderlich.

ToByteArray()

Um Daten über eine Socket-Verbindung schicken zu können, müssen diese als Bytes vorhanden sein. Dazu wird diese Funktion genutzt. Sie wandelt einen übergebenen String aus Hexadezimalzahlen in ein Bytearray um. Die Anzahl der benötigten Bytes ist gleich der halben Länge des Strings. Für die Umwandlung eines Strings in ein Byte bietet C# die Funktion „Convert.ToByteArray()“ an. Diese Funktion bekommt einen String aus zwei Zeichen und die Zahlenbasis, für Hexadezimalzahlen 16, übergeben und liefert ein Byte zurück. Folglich werden jeweils zwei aufeinanderfolgende Zeichen des Strings zu einem Byte zusammengefasst und nacheinander in ein Array aus Bytes geschrieben. Sind alle Zeichen umgewandelt, dient das Bytearray als Rückgabewert.

BCC()

Für jedes zu sendende Telegramm wird ein „Block Check Charakter“ benötigt. Dieser wird als Exklusiv-Oder Verknüpfung der Daten berechnet. Für das Schreiben eines Parameters werden die Zeichen des Telegramms aus Abbildung 33, ab dem Dienst bis inklusive dem Steuerzeichen berücksichtigt. Die notwendigen Daten werden in Form eines Strings übergeben, wobei das Steuerzeichen als eigener Parameter benötigt wird. Danach wird der String in ein Charakterarray umgewandelt und die einzelnen Zeichen miteinander Exklusiv-Oder verknüpft. Anschließend wird dem Steuerzeichen sein entsprechender Hexadezimalwert zugeordnet und mit dem bereits vorhandenen Wert verknüpft. Am Schluss wird Überprüft, ob der „Block Check Charakter“ größer als 20h ist. Falls dies nicht der Fall ist wird er um diesen Wert erhöht und dient als Rückgabewert der Funktion.

AsciiWert()

Diese Funktion wandelt einen String in die entsprechenden ASCII-Werte um. Dazu wird der übergebene String in ein Charakterarray umgewandelt. Anschließend werden den einzelnen Zeichen des Feldes mit der C# Funktion „Convert.ToInt32()“ der ASCII-Wert zugeordnet. Danach erfolgt die Umrechnung dieser Werte in Hexadezimalzahlen, welche nacheinander in einen Rückgabestring geschrieben werden.

CheckIID()

Diese Methode überprüft, ob die „Invoke-ID“, welche nach jedem Telegramm an den Frequenzumrichter erhöht wird, kleiner als 15 ist. Falls nicht, wird sie auf den Wert 1 gesetzt, damit die Kommunikation fortgesetzt werden kann, da ein Wert größer 15 ein Fehler verursacht. Deshalb wird diese Funktion nach jedem Telegramm an den Frequenzumrichter aufgerufen.

Steuerzeichen()

Die Response des Frequenzumrichters wird immer mit einem Steuerzeichen abgeschlossen. Um dieses Auswerten zu können wird dem empfangenen Wert, entsprechend Tabelle 1, ein String zugeordnet.

GetHex()

Für die Kommunikation mit dem Frequenzumrichter werden Hexadezimalwerte in Form von Charakter oder Strings benötigt. Diese Funktion bekommt eine Dezimalzahl, im Bereich von 0 bis 15, übergeben und liefert ihren Hexadezimalwert in Form eines Strings zurück.

GetDezZeichen()

Diese Funktion wird für die Auswertung der empfangenen Daten benötigt und weist einem empfangenen, hexadezimalen Zeichen, welches als String übergeben wird, einen ganzzahligen Dezimalwert zu.

DezAusHex()

Wenn Parameterwerte von einem Frequenzumrichter gelesen werden, so sind diese, in den empfangenen Telegrammen, in hexadezimaler Form vorhanden. Zusätzlich werden die Daten als Strings empfangen, wodurch diese erst in eine Dezimalzahl umgewandelt werden müssen. Dazu werden die Strings Zeichen für Zeichen mit der Funktion „DezAusHex()“ in Dezimalzahlen umgewandelt, mit 16 multipliziert und aufsummiert. Das Ergebnis der Berechnung ist der empfangene Parameterwert in Form einer dezimalen ganzen Zahl und dient als Rückgabewert dieser Funktion.

Fill8()

Für das Schreiben von Parameterwerten auf den Frequenzumrichter müssen diese, wie in Abbildung 33 ersichtlich, acht Zeichen lang sein. Dafür wurde diese Funktion implementiert, welche eine Dezimalzahl in einen acht Zeichen langen String aus Hexadezimalzeichen umrechnet. Zuerst werden die Stellen der Hexadezimalzahl mit fallender Wertigkeit berechnet und mit der Funktion „GetHex()“ in Hexadezimalzahlen umgewandelt. Anschließend werden die Zeichen in einem String aneinandergefügt. Ist der String kürzer als 8 Zeichen, werden die höherwertigen freien Stellen mit Nullen aufgefüllt.

BefehlParameterSchreiben()

Für das Schreiben von Parametern auf den Frequenzumrichter sind Telegramme wie in Abbildung 33 erforderlich. Diese Funktion bekommt die Adresse, den Parameterwert, die Satzkennung sowie die aktuelle „Invoke-ID“ übergeben und stellt daraus das Telegramm in Form eines Strings zusammen.
Dazu wird zuerst ein leerer String mit dem Namen „Befehl“ erzeugt. Danach wird der Parameter mittels der Funktion „Fill8()“ in einen acht Zeichen langen String umgewandelt, wobei zusätzlich der Dezimalwert auf hexadezimal umgerechnet wird.

Als nächster Schritt wird der String „Befehl“ mit einem „H“, für die Bezeichnung des Dienstes, der Adresse, dem Parameterstring und der Satznummer beschrieben. Die Funktion „BCC()“ berechnet aus diesem String und dem Steuerzeichen „ETX“ den „Block Check Charakter“. Der Befehlsstring wird mittels der Funktion „AsciiWert()“ zeichenweise in die hexadezimalen ASCII-Werte konvertiert. Zur Vervollständigung des Telegramms werden noch die Steuerzeichen, am Anfang und am Ende, sowie der „Block Check Charakter“ angefügt. Somit ist der Datenrahmen fertig.

Senden()

Die Funktion Senden bekommt das fertige Telegramm, als String, übergeben und wandelt es mittels der Funktion „ToByteArray()“ in ein Bytearray um. Für das Senden der Bytes über die existierende Socket-Verbindung wir der C# Befehl „Socket.Send()“ benutzt, wobei „Socket“ für den Socket einer existierenden Verbindung steht. Die Parameter dieser Funktion sind das Bytearray, dessen Länge sowie die Socket Flags. Ist ein Senden nicht möglich wird eine Fehlermeldung mittels einer Exception erzeugt und an die nächste höhere Programmebene geschickt.

Empfangen()

Da die Kommunikation zwischen zwei Kommunikationsendpunkten über Datenströme aus Bytes erfolgt, werden mittels dieser Funktion, die empfangenen Bytes decodiert und in einen String umgewandelt. Zuerst wird der Bytestrom von dem Socket gelesen, die Anzahl der empfangenen Bytes bestimmt und mittels eines Encoders in ein Charakterarray umgewandelt. Dieses wird anschließend noch in einen String umgeformt, der als Rückgabewert dient.

ParameterLesen()

Diese Funktion dient zum Lesen eines Parameters von einem Frequenzumrichter. Die Übergabeparameter sind die Parameteradresse, die Nummer des benutzten Satzes sowie die „Invoke-ID“. Als erster Schritt erfolgt die Überprüfung, ob eine Socket-Verbindung vorhanden ist, falls nicht wird eine Fehlermeldung generiert. Anschließend wird ein Telegramm wie in Abbildung 36 erzeugt. Dazu wird ein leerer String benötigt, und dieser anschließend mit einem „H“, dem mittels der Funktion „GetHex()“ in eine Hexadezimalzahl umgewandelten „Invoke-ID“, der Parameteradresse und der Satznummer beschrieben. Von diesem String wird mit der Funktion „BCC()“ der „Block Check Charakter“ berechnet. Die Funktion „AsciiWert()“ wandelt die Zeichen des Strings in die hexadezimalen ASCII-Werte um. Damit das Telegramm vollständig ist, wird am Ende des Befehls noch 05h für das Steuerzeichen „ENQ“ und der „Block Check Charakter“ angehängt.
Durch den Aufruf der Funktion „Senden()“ wird der Befehl an den Frequenzumrichter geschickt und dessen Antwort mittels der Funktion „Empfangen()“ entgegengenommen. Da hier nur Telegramme mit Parameterwerten behandelt werden, erfolgt die Auswertung über die Länge des empfangen Strings. Werden weniger als 13 Zeichen empfangen, wird eine Fehlermeldung mittels einer Exception an die nächste Programmebene weitergereicht.
Handelt es sich bei den empfangenen Daten um einen fehlerfreien Datenrahmen, so wird der Parameterwert durch Abschneiden der ersten drei Zeichen am Anfang beziehungsweise der letzten beiden Zeichen gewonnen. Anschließend werden noch alle führenden Nullen entfernt und der Parameterwert mittels der Funktion „DezAusHex()“ in eine Dezimalzahl umgerechnet. Dieser Wert dient als Rückgabewert der Funktion „ParameterLesen()“.

4.2.3. Unterprogramme der Motorsteuerung

Für die Steuerungen der Servomotoren sind Unterprogramme notwendig, welche die notwendigen Parameter verändern. Es gibt zwei Betriebsmodi, einen in dem der Servomotor immer nur einem Positionsindex folgt und den zweiten, in dem zwei Indizes benützt werden und somit der Ausleger zwischen zwei Positionen hin und her schwingt. Deshalb wurden für beide Modi Funktionen für die Initialisierung, das Ändern der Geschwindigkeit und das Ändern der Position implementiert.
Um die unterschiedlichen Betriebsmodi besser darstellen zu können ist in Abbildung 41 die Positionierung mit nur einem Positionsindex dargestellt. Der Servomotor hält eine Position so lange bis ein neuer Parameterwert übergeben wird.

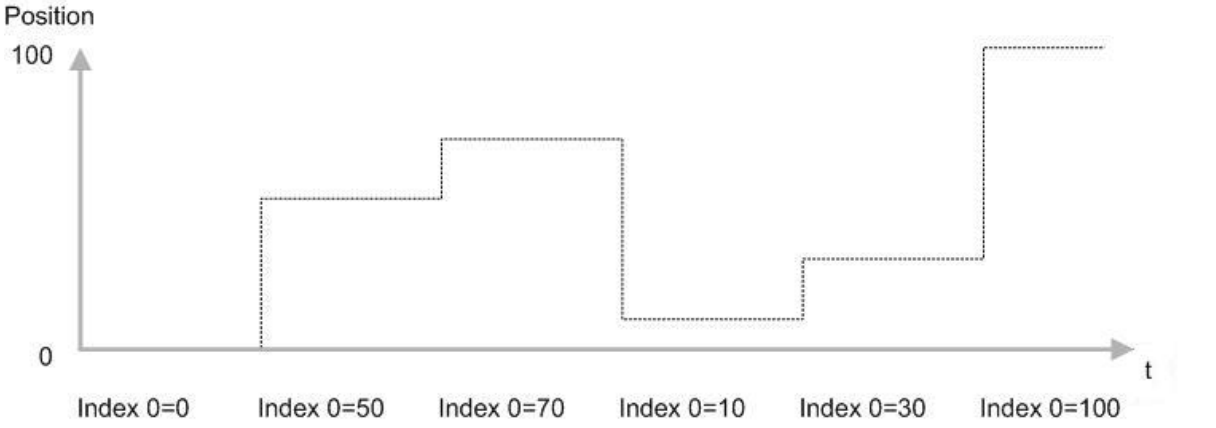


Abbildung 41: Verwendung eines Positionsindex

Im Gegensatz dazu kann bei der Verwendung von mehreren Indizes zwischen diesen hin und her geschalten werden. Abbildung 42 zeigt die Verwendung von zwei Indizes wobei Positionsindex 1 fest auf den Wert 100 steht und Positionsindex 0 variabel ist. Dabei wurde festgelegt, dass Index 0 auf Index 1 folgt und umgekehrt. Würde man den Folgeindex nicht angeben, dann würde der Frequenzumrichter nicht auf den nächsten Index wechseln.

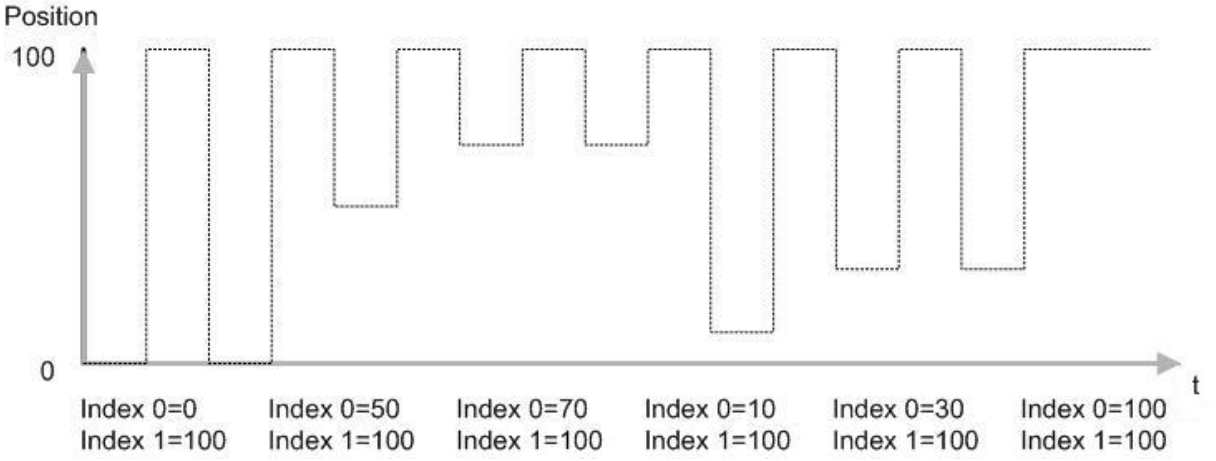


Abbildung 42: Verwendung zweier Positionsindizes, wobei Index 1 fix ist

Ebenso ist es möglich, wie in Abbildung 43 beide Positionsindizes zu verändern.

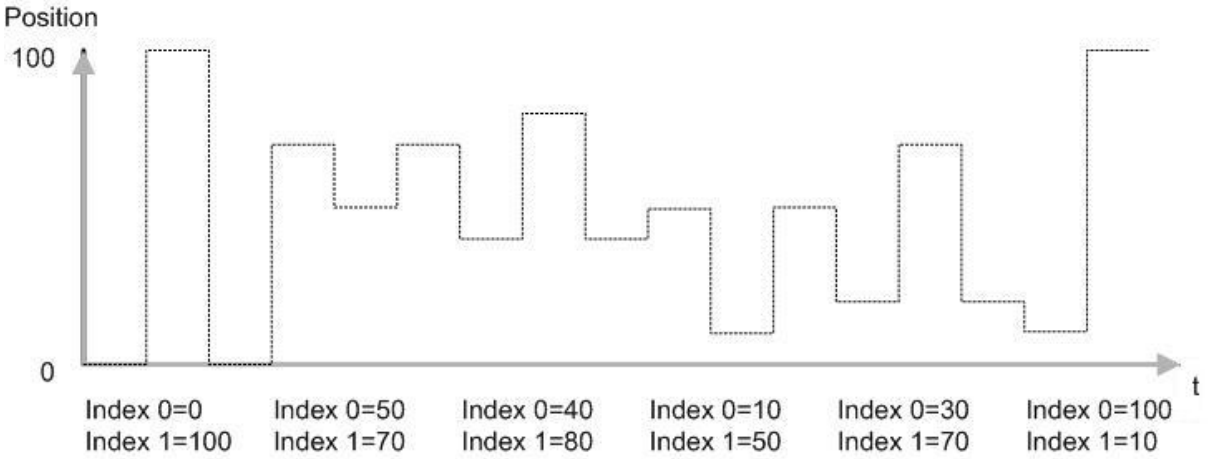


Abbildung 43: Verwendung zweier Positionsindizes

Da man zwischen mehreren Indizes wählen kann, kann für jeden Index die Position, Geschwindigkeit und die Nummer des nachfolgenden Index bestimmt werden, wie man in Abbildung 28 oder auch in der Parameterliste aus Abbildung 31 sehen kann.
Um einen Parameterwert auf den Frequenzumrichter zu schreiben, wurde wie in Abbildung 44 vorgegangen. Zuerst wurde das Telegramm, welches durch die Variable vom Typ String mit dem Namen Befehl dargestellt wird, mittels der Funktion „BefehlParameterSchreiben()“ generiert. Die dafür benötigte Adresse des Parameters ist in der Combivis5 Software zu finden, beispielsweise wenn man den Parameter auf eine Parameterliste setzt (siehe Abbildung 31). Die Funktion „BefehlParameterSchreiben()“ benötigt für das Telegramm noch den Parameterwert, die Satzkennung und die „Invoke-ID“. Dieser Befehl wird dann mittels der Funktion „Senden()“ über die aktuelle Socket-Verbindung geschickt. Die Response des Frequenzumrichters wird mittels der Funktion „Empfangen()“ von einem Bytestrom in den String Empfang umgewandelt. Die Funktion „ResponseParameterSchreiben()“ wertet die empfangenen Daten aus und liefert im fehlerfreien Fall den Wert „true“ zurück. Ist dies der Fall, so wird noch die Invoke-ID mittels „CheckIID()“ überprüft und gegebenenfalls korrigiert.
Wurde jedoch von Frequenzumrichter eine negative Quittung geschickt, dann wird die Verbindung zum Frequenzumrichter getrennt und eine FormatException an die übergeordnete Programmebene geschickt. Bei der Fehlermeldung wir angegeben, welcher Parameter nicht geschrieben werden konnte.

```
Befehl = BefehlParameterSchreiben(Adresse, Parameterwert, Satz, IID);
Senden(Befehl);
Empfang = Empfangen();
if(ResponseParameterSchreiben(Empfang,IID)= =false)
{
        Trennen();
        Throw new FormatException(„ Fehler bei …");
}
IID = CheckIID(IID);
```

Abbildung 44: Schreiben eines Parameterwertes

Jedes der Unterprogramme besteht aus mehreren von den in Abbildung 44 dargestellten Blöcken. Der grundlegende Aufbau der Unterprogramme zeigt Abbildung 45. Dem Unterprogramm werden die IP-Adresse, die Satzkennung, die aktuelle „Invoke-ID“ und die benötigten Parameter, deren Anzahl je nach Funktion unterschiedlich ist, übergeben. Zuerst wird die Verbindung zu einem Frequenzumrichter aufgebaut, wobei die Funktion „Verbinden()“ in einem try-catch Block steht die im Fehlerfall eine Fehlermeldung über eine MessageBox ausgibt.
War der Verbindungsaufbau erfolgreich, folgen die Blöcke aus Abbildung 44 zum Schreiben der Parameterwerte an den Frequenzumrichter. Auch für diese Blöcke erfolgt eine Ausnahmebehandlung mittels eines try-catch Blockes, der beim Auftreten einer FormatException eine Fehlermeldung am Bildschirm ausgibt. Der Text der Fehlermeldung setzt sich aus dem Parameter, der nicht geschrieben werden konnte, und dem Namen des Unterprogramms zusammen, siehe Abbildung 46.

```
public int Unterprogramm(IP-Adresse, Parameter1,...,Satzkennung, IID)
{
        try
        {
                Verbinden(IP-Adresse)
        }
        catch(SocketException se)
        {
                Fehlerbehandlung: Exception weiterreichen oder Fehlermeldung
        }
        try
        {
                Block Parameter1 schreiben

                Block Parameter2 schreiben
        }
        catch(FormatException fe)
        {
                Fehlerbehandlung: Exception weiterreichen oder Fehlermeldung
        }

}
```

Abbildung 45: Grundlegender Aufbau eines Unterprogramms

Abbildung 46 zeigt eine Fehlermeldung, welche vom Hauptprogramm in Form einer MessageBox ausgegeben wird. In der ersten Zeile steht in welchem Unterprogramm ein Fehler aufgetreten ist. Die zweite Zeile gibt an, welcher Fehler genau aufgetreten ist. Hier konnte der Parameterwert für den Parameter „Posi/Synchronmodus“ nicht geschrieben werden und deshalb wurde die Initialisierung abgebrochen.

Abbildung 46: Fehlermeldung

Grundsätzlich ist es möglich den Frequenzumrichter mit mehreren Sätzen zu betreiben. Deshalb wird die Satzkennung auch jedem Unterprogramm beziehungsweise jeder Funktion übergeben, die mit dem Umrichter kommuniziert. Für diese Anwendung wird nur ein Satz benötigt, deshalb wurde die Satzkennung als globale Variable definiert und mit dem Wert 0 initialisiert.
Die für die Programmierung notwendigen Parameter und Parameterwerte wurden mittels der Combivis5 Software ermittelt und getestet. Für den Betrieb der Frequenzumrichter im „Posimodus“ sind die Parameter aus Abbildung 28 ausreichend und darauf beziehen sich auch die Bezeichnungen im unteren Abschnitt.

Unterprogramme für die Positionierung mit einem Index

Für die folgenden Unterprogramme wird nur der Index 0 zur Positionierung verwendet, was eine Positionierung wie in Abbildung 41 zur Folge hat. Aus diesem Grund müssen auch nur die Parameterwerte für diesen Index verändert werden.

NeuePositionSchreiben()

Um die Position des Servomotors verändern zu können, benötigt dieses Unterprogramm als Übergabeparameter die IP-Adresse des Frequenzumrichters, die neue Position, die Satzkennung und die „Invoke-ID“. Wie in Abbildung 45 dargestellt erfolgt zuerst der Verbindungsaufbau zu dem entsprechenden Umrichter durch den Aufruf der Funktion „Verbinden()“. Wird keine SocketException empfangen, dann wird auf den Parameter „Index/Auswahl“ der Wert 0 geschrieben und anschließend erfolgt das Senden des neuen Positionswertes auf „Index/Position“. Das Schreiben der Parameterwerte erfolgt mittels der in Abbildung 44 dargestellten Blöcke. Für ein Ändern der Position ist es nicht notwendig die Reglerfreigabe zu betätigen, dies kann auch während des laufenden Betriebes erfolgen.

EinIndexGeschwindigkeitSchreiben()

Dieses Unterprogramm dient zur Geschwindigkeitsänderung des Positioniervorganges. Als Übergabeparameter werden die IP-Adresse, die neue Geschwindigkeit, die Satzkennung sowie die „Invoke-ID benötigt“. Nach dem Verbindungsaufbau erfolgt das Schreiben des Parameterwertes 0 für den Parameter „Index/Auswahl“, sowie der neuen Geschwindigkeit für den Parameter „Index/Geschwindigkeit“. Auch die Änderung der Positioniergeschwindigkeit kann ohne Betätigung der Reglerfreigabe erfolgen.

Init()

Ähnlich der Parameterliste aus Abbildung 31 dient dieses Unterprogramm zur Initialisierung des Positionierbetriebes. Dafür sind IP-Adresse, die Satzkennung, die „Invoke-ID“ sowie die Parameterwerte für die Position und die Geschwindigkeit erforderlich. Nach dem Verbindungsaufbau wird auf den Parameter „Index/Modus“ der Wert 1 geschrieben. Das bedeutet, dass eine Fortsetzung der Profilabarbeitung möglich ist und die Positionsvorgabe absolut erfolgt. Die Ermittlung des Parameterwertes für „Index/Modus“ erfolgte mit der

Combivis5 Software und wird für diesen Parameter in Abbildung 30 gezeigt. Als nächstes wird auf den Parameter „Index/Auswahl“ der Wert 0 geschrieben. Anschließend werden auf die Parameter „Index/Geschwindigkeit“ und „Index/Position“, für den Index 0, die Parameterwerte geschrieben. Zuletzt wird noch der Parameter „Index/Nächster“ auf den Wert 0 gesetzt, da nur dieser Index für die Positionierung verwendet wird.

Unterprogramme für die Positionierung mit zwei Indizes

Für die Positionierung mit zwei Positionsindizes sind eigene Unterprogramme notwenig, die sich geringfügig von den oben beschriebenen Programmen unterscheiden, da die Parameter für die beiden benutzten Indizes verändert werden müssen. Es werden die Indizes 0 und 1 für den Positioniervorgang verwendet.

IndexPositionSchreiben()

Dieses Unterprogramm hat im Gegensatz zu „NeuePositionSchreiben()“ einen Übergabeparameter mehr. Zusätzlich zur IP-Adresse des Frequenzumrichters, der neuen Position, der Satzkennung und der „Invoke-ID“ wird hier noch der Index benötigt, dessen Position geändert werden soll. Dadurch ist es einerseits möglich den Ausweichvorgang auszuführen und andererseits kann auch eingestellt werden, bis zu welcher Position die Schwingbewegung durchgeführt wird. Da die Zeichenplatte schmäler als der Bildbereich ist und wenn die Ausleger in Fahrtrichtung ausgerichtet sind, dann ragen die Stifte über den Rand der Zeichenplatte hinaus und beschreiben den Untergrund. Deshalb müssen die äußeren Grenzen der Schwingbewegung einstellbar sein.
Nach dem Verbindungsaufbau mit dem entsprechenden Frequenzumrichter, wird auf den Parameter „Index/Auswahl“ der Wert des Index geschrieben, der geändert werden soll. Im Anschluss wird auf den Parameter „Index/Position“ die neue Position eingetragen.

IndexGeschwindigkeitSchreiben()

Dieses Unterprogramm dient zur Änderung der Geschwindigkeit für die Positionierung. Es ist möglich für jeden Index eine eigene Geschwindigkeit festzulegen. Da dies für die Anwendung nicht erforderlich ist, werden für beide Indizes die gleiche Geschwindigkeit genutzt. Deshalb werden dieselben Übergabeparameter wie bei dem Unterprogramm „EinIndexGeschwindigkeitSchreiben()“ benötigt.
Auf den Verbindungsaufbau erfolgt das Schreiben des Parameterwertes 0 für den Parameter „Index/Auswahl“, sowie das Schreiben des Parameterwertes für den Parameter „Index/Geschwindigkeit“. Dieser Vorgang wird für den Index 1 wiederholt.

Initialisierung()

So wie die Funktion „Init()“ dient dieses Unterprogramm dazu, die Art der Positionierung festzulegen. Da hier zwei Indizes benutzt werden wird nach Ablauf dieses Programms der Servomotor mit der übergeben Geschwindigkeit zwischen den beiden Positionen hin und her wechseln.
Als Übergabeparameter werden die IP-Adresse des Frequenzumrichters, die beiden Positionen, die Geschwindigkeit und die „Invoke-ID“ benötigt. Der erste Schritt der Initialisierung ist das Ausführen des Unterprogramms „IndexGeschwindigkeitSchreiben()“. Anschließend erfolgt der Verbindungsaufbau und der Parameter Index/Modus wird, wie auch in Unterprogramm „Init()“, auf den Wert 1 gesetzt. Im Anschluss daran erfolgt über den Parameter „Index/Auswahl“ die Auswahl des Index 0, dessen „Index/Position“ geschrieben und der Parameter „Index/Nächster“ auf den Wert 1 gesetzt wird. Anschließend wird dieser Vorgang für den Index 1 wiederholt, jedoch wird als nachfolgender Index der Index 0 festgelegt. Dies hat zur Folge, dass immer zwischen den Parametern von Index 0 und Index 1

gewechselt wird. Unterschiedliche „Index/Geschwindigkeit“ Parameter würden sich dann bemerkbar machen.

Stoppen der Motoren

Da die Servomotoren bei der Positionierung Geschwindigkeiten bis 4000 Umdrehungen pro Minute realisieren können, ist ein sicheres Ausschalten der Motoren wichtig. Die Reglerfreigabe steuert die Endstufen des Frequenzumrichters an [6]. Wird sie eingeschaltet, dann beginnen die Umrichter das gespeicherte Programm auszuführen. Beim Ausschalten der Reglerfreigabe, wird das Programm sofort unterbrochen und die Servomotoren nicht mehr geregelt, jedoch nicht abgebremst. Dies hat zur Folge, dass bei einer Positionierung mit hoher Geschwindigkeit der Motor, nach dem Abschalten der Reglerfreigabe sich weiterdreht und langsam ausläuft. Dieses Verhalten ist unerwünscht, da das langsame Auslaufen dazu führt, dass die Ausleger des Demonstrators mit den Rollen des Versuchaufbaus kollidieren. Deshalb wurde eine Funktion implementiert, die den Motor sicher abbremst. Dabei ist es unerheblich ob die Positionierung mit einem oder zwei Indizes erfolgt.

StopIndex()

Diese Funktion benötigt als Übergabeparameter die IP-Adresse, die Position sowie die „Invoke-ID“. Das Stoppen der Servomotoren ist vor allem beim Betrieb mit zwei Indizes wichtig, da in diesem Betriebsmodus ein starkes Schwingen der Ausleger erzeugt wird. Bei der Positionierung mit einem Index erfolgt eine Beschleunigung der Ausleger nur am Anfang und Ende eines Ausweichvorganges. Bei einem Abschalten der Steuerung zu diesen Zeitpunkten, würde die Bewegung der Ausleger unkontrolliert weitergeführt werden. Wohingegen ein Abschalten zu einem Zeitpunkt, wo keine Positionsänderung erfolgt, nur einen kleinen Ausschlag des Auslegers zur Folge hat.
Deshalb wird für das Stoppen der Servomotoren zuerst auf den Parameter „Index/Auswahl“ der Wert 0 geschrieben. Anschließend wird die übergebene Position, für die Endstellung der Ausleger, auf den Parameter „Index/Position“ geschrieben und der Parameter „Index/Nächster“ ebenso auf den Wert 0 gesetzt. Dadurch verharrt der Servomotor auf der vorgegeben Position, beziehungsweise bewegt sich der Servomotor in einem kleinen Bereich um diesen Wert, je nachdem wie stark dessen Welle belastet ist, da der Regler des Frequenzumrichters dadurch nicht genau auf diese Position regeln kann. Im Anschluss daran wird die Index/Geschwindigkeit auf den Wert 0 gesetzt. Wird nach Ausführung dieser Funktion die Reglerfreigabe deaktiviert, dann ändert sich die Position des Motors nicht mehr.

5. Bildverarbeitungssoftware

Aufgabe der Bildverarbeitungssoftware ist es, ein Bild aufzunehmen, die Objekte zu extrahieren und deren Position so lange zu Verfolgen beziehungsweise zu berechnen, bis sie den Zeichenbereich der Motoren verlassen haben. Das Ziel ist, die Objektpositionen möglichst genau berechnen zu können und deshalb ist es erforderlich möglichst viele Bilder innerhalb kürzester Zeit aufzunehmen und zu verarbeiten. Dadurch wird eine möglichst kleine örtliche Verschiebung zwischen zwei aufeinander folgender Bilder erreicht. Würde die Verschiebung, in Fahrtrichtung, zwischen zwei Bildern größer sein, als der Abstand zwischen zwei Objekten, so können diese nicht mehr einander zugeordnet werden. Andererseits ist es für die Entwicklung und im Fehlerfall notwendig, die Resultate der einzelnen Bildverarbeitungsschritte zu begutachten, weshalb das Speichern von Ergebnisse möglich sein muss. Für das schnelle Bearbeiten der Bilder werden diese in Matrizen umgewandelt, da der Zugriff auf einzelne Bildpunkte mittels den C# Funktionen „GetPixel()“ und „SetPixel()“ verhältnismäßig langsam erfolgt. Der Vorteil an Geschwindigkeit bringt leider den Nachteil mit sich, dass die Resultate nur schwer darstellbar sind. Aus diesem Grund wurden die Funktionen „Otsu()“, „ComponentLabeling()“ und „Registrierung()“ so implementiert, dass sie als Übergabeparameter einen Dateinamen benötigen. Wird eine Funktion mit einem leeren String als Dateiname aufgerufen, so erfolgt die Berechnung mit den Matrizen. Wird hingegen ein Pfad, als Ziel für das Speichern, übergeben, so wird am Ende der Berechnung ein Bild aus der Matrix erstellt und gespeichert. Somit ist es einerseits möglich sehr schnell mit den Matrizen zu arbeiten, andererseits können die Resultate einzelner oder auch aller Bildverarbeitungsschritte gespeichert werden.

5.1. Otsu()

Diese Funktion ist der erste Bildverarbeitungsschritt nach der Aufnahme des Bildes. Beim Aufruf dieser Funktion dient das Bild sowie ein String für den Speicherort als Übergabeparameter. Der Rückgabeparameter ist eine Binärmatrix. Da es für das Component-Labeling Voraussetzung ist, dass sich keine Objekte am Bildrand befinden, wird bei der Erzeugung der Binärmatirx ein weißer Rand um das binarisierte Bild gezogen. Dadurch vergrößert sich die Dimension der Matrix im Vergleich zum Bild um zwei Pixel in der Länge und der Breite, jedoch erspart man sich einige Abfragen während des Segmentierungsvorganges. Um die Abmessungen des Bildes zu erhalten, gibt es bei C# die Möglichkeit mittels „Bitmap.Height()“ und „Bitmap.Widht()“ diese Parameter aus dem übergebenen Bild zu ermitteln. Dies bietet die Möglichkeit diese Funktion und auch alle weiteren Bildverarbeitungsschritte auf Bilder unterschiedlicher Größe anzuwenden.
In Abbildung 47 ist der Ablauf der Funktion illustriert. Für die Bildaufnahme wird eine Farbkamera benutzt, weshalb aus dem Farbbild für die weitere Verarbeitung in einem ersten Schritt ein Grauwertbild erzeugt wird. Die C# Funktion „GetPixel()“ beziehungsweise „SetPixel()“ sind für diesen Vorgang zu langsam, weshalb die Farbwerte mittels Zeigern direkt aus dem Bild ausgelesen werden. Die Funktion „PixelFormat()“ liefert die Informationen wie die Farbwerte der einzelnen Bilder gespeichert werden. Das Format eines Bildpunktes ist für die Berechnung des Zeigers notwendig, da unterschiedlich viele Bytes für die Farbcodierung verwendet werden. Dadurch wird sichergestellt, dass der Zeiger auch auf die richtigen Werte im Bild verweißt. Nun können der Rot-, Grün- und Blauwert ausgelesen und der Grauwert berechnet werden. Anschließend wird der Grauwert in eine Matrix eingetragen, wobei die Koordinaten des Bildpunktes den Indizes des Matrixelements entsprechen. Die Matrix stimmt dann mit dem Graustufenbild überein, wobei hier bereits der weiße Rand berücksichtigt wird. Für das Verfahren von Otsu wird ein Grauwerthistogramm benötigt, welches gleichzeitig mit der Umwandlung in ein Grauwertbild erstellt wird. Dazu

wird eine leere Liste mit 256 Indizes erzeugt, wobei jede Stelle für einen Grauwert steht. Jeder berechnete Grauwert erhöht den Wert an dem entsprechenden Index und nachdem das Bild vollständig umgewandelt wurde, enthält die Liste das vollständige Histogramm des Bildes.

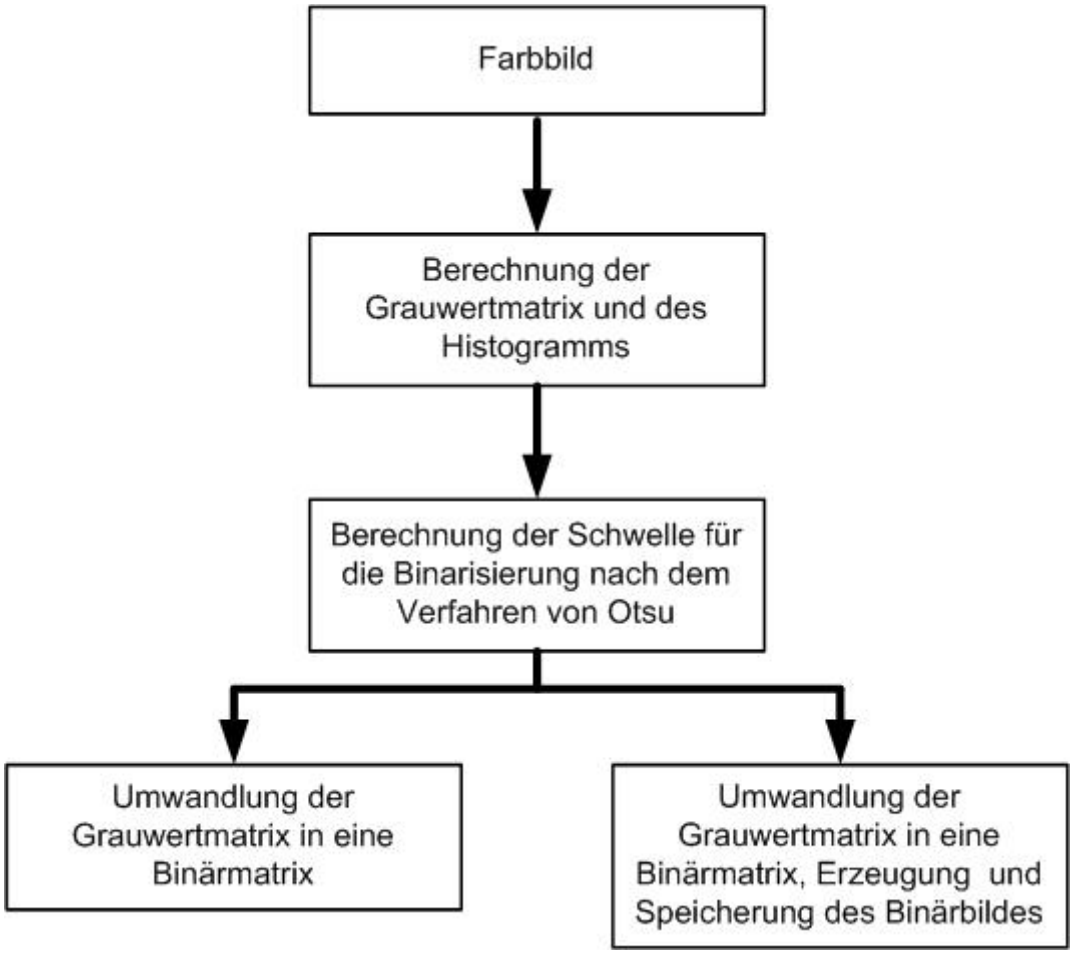


Abbildung 47: Otsu Verfahren im Überblick

Nachdem das Histogramm erstellt worden ist, erfolgt die Berechnung der Schwelle für die Binarisierung mittels der in Kapitel 2.2 beschriebenen Methode. Im Anschluss daran erfolgt die Überprüfung, ob es sich bei dem Übergabeparameter für den Pfad um einen leeren String handelt oder nicht. Wird kein Pfad übergeben, dann wird jedes Element der Grauwertmatrix mit dem Schwellwert verglichen. Ist der Grauwert größer als die Schwelle, dann wird das Element auf 0 gesetzt, was der Farbe Weiß entspricht. Anderenfalls wird das Element auf 1 gesetzt, was für die Farbe Schwarz steht.
Wenn jedoch ein Pfad zum Speichern des Binärbildes übergeben wird, dann wird ein neues Bitmap erzeugt. Zusätzlich zum Binarisieren der Matrixelemente werden schwarze oder weiße Bildpunkte in das neue Bild eingezeichnet. Das Binärbild wird dann unter dem angegebenen Pfad gespeichert und vor der Extension wird _o eingefügt um zu signalisieren, dass dieses Bild von der Funktion „Otsu()“ erzeugt wurde.

5.2. Component-Labeling()

Hierbei handelt es sich um die Implementierung des Component-Labeling Algorithmus aus Kapitel 2.3.1. Einziger Unterschied ist, dass keine zusätzliche Matrix zum Speichern der Segmentkonturen implementiert wurde, da diese nicht benötigt wird.
Die Segmentierung benötigt als Eingangsvariable die Binärmatrix und liefert als Ausgang eine Matrix mit den markierten Segmenten, sowie deren Anzahl. Somit setzt sich die Funktion „Component-Labeling()“ aus vier Programmteilen zusammen:

5.2.1. Tracer()

Die Funktion „Tracer()“ wird dazu verwendet, um bei der Konturverfolgung den nächsten schwarzen Bildpunkt zu finden. Sie bekommt die Binärmatrix, die Matrix zum Markieren der Segmente sowie die Liste „cxy“ übergeben. Die Elemente der Liste „cxy“ sind die x- und y-Koordinate der aktuellen Position sowie der neue Suchindex. Dieser Index bestimmt den Punkt in der Nachbarschaft der aktuellen Position, bei dem die Suche gestartet wird. In Abbildung 48 ist die Nachbarschaft mit den Indizes und Richtungen des nächsten Nachbarn um die aktuelle Position (x,y) dargestellt. Die Position des nächsten Matrixelements, welches überprüft wird, errechnet sich aus den aktuellen Koordinaten plus dem Vektor des Suchindex.

5 (-1,-1)	6 (0,-1)	7 (1,-1)
4 (-1,0)	(x,y)	0 (1,0)
3 (-1,1)	2 (0,1)	1 (1,1)

Abbildung 48: Indizes und Richtung um die aktuelle Position (x,y)

Handelt es sich bei diesem Matrixelement um eine 0, was einem weißen Bildpunkt entspricht, so ist dies ein Teil des Randes um das Segment und wird mit einer -1 markiert. Anschließend wird die Suche im Uhrzeigersinn fortgesetzt. Kann in der Nachbarschaft kein schwarzer Bildpunkt gefunden werden, dann handelt es sich um einen einzelnen Punkt und die Suche wird abgebrochen. Die Position wird unverändert an den aufrufenden Programmteil zurückgeliefert.
Ist das gefundene Matrixelement jedoch eine 1, was einem schwarzen Bildpunkt entspricht, so werden dessen Position sowie der Suchindex in die Liste „cxy“ geschrieben und zum aufrufenden Programm zurückgekehrt. Da die Koordinaten und der Suchindex in einer Liste gespeichert werden, müssen sie nicht explizit als Rückgabewerte der Funktion definiert werden um dem Programmteil, der die Funktion „Tracer()“ aufgerufen hat, zur Verfügung zu stehen.

5.2.2. ContourTracing()

Wird beim Durchlauf durch das Binärbild, eigentlich die Matrix die diesem Bild entspricht, ein schwarzer Bildpunkt gefunden welcher der Startpunkt einer neuen Kontur ist, so wird diese Funktion zur Konturverfolgung aufgerufen. Beim Aufruf werden die Binärmatrix, die Matrix mit den markierten Segmenten, die Nummer des neuen Labels und die Liste „cxy“, welche die Koordinaten des Bildpunktes sowie den Index für die Suche enthält, übergeben.

Bei der Konturverfolgung werden zuerst die x- und y-Koordinate von „cxy“ als Startpunkt gespeichert und dienen, wenn die Kontur vollkommen durchlaufen wurde, als Abbruchbedingung. Deshalb ist es auch möglich einen isolierten Punkt zu erkennen, da die Funktion „Tracer()“ nur die Werte von „cxy“ verändert, wenn ein Nachbar gefunden wurde. Der von der Funktion „Tracer()“ zurück gelieferte Punkt wird in der Matrix für die Segmentierung mit dem übergebenen Label markiert und danach der Index für die neue Suche bestimmt. Dazu wird die Formel

$$neueSuchrichtung = (alteSuchrichtung + 6)\%8 \quad \textbf{(5.1)}$$

verwendet. Dieser Vorgang wird so lange wiederholt, bis die Startposition wieder erreicht wird. Somit ist die Kontur vollständig markiert, unabhängig davon, ob es sich um eine äußere oder innere Kontur handelt.

5.2.3. ConnectedComponentLabeling()

Hierbei handelt es sich um die Grundfunktion zum Markieren der Segmente. Dazu wird die Binärmatrix benötigt und eine Matrix mit den markierten Segmenten sowie deren Anzahl zurückgeliefert. Wird bei einem Aufruf der Funktion auch ein Pfad übergeben, dann wird die Funktion „BildErstellen()“ aufgerufen, welche ein Bild mit den markierten Regionen erstellt.
Die Binärmatrix wird elmenteweise von links oben nach rechts unten durchlaufen. Wobei den Werten „cx“ und „cy“ die Indizes der Matrix, beziehungsweise den Koordinaten der Bildpunkte entsprechen. Zusätzlich gibt es noch eine Variable, welche die Anzahl der Label zählt und eine andere die das zur Markierung benötigte Label enthält. Am Anfang wird eine Matrix in Größe der Binärmatrix erstellt, welche für die Segmentierung genutzt wird. Diese Matrix wird dann am Ende des Segmentierungsvorganges in die Ausgabematrix kopiert. Dieser Schritt ist notwendig, da dadurch eine neue Binärmatrix segmentiert werden kann und gleichzeitig die Ausgabematrix einer anderen Funktion zur Verfügung steht.
Für die Segmentierung wird die Binärmatrix elementweise durchlaufen. Wird ein unmarkiertes schwarzes Element, repräsentiert durch eine 1, gefunden, dann handelt es sich um ein neues Segment. Die Koordinaten des Elements entsprechen dem Startpunkt einer neuen, äußeren Kontur und werden in die Liste „cxy“ geschrieben. Der Suchindex wird in diesem Fall auf den Wert 0 gesetzt, da die Matrix zeilenweise durchlaufen wird. Die Liste „cxy“ dient bei einem Aufruf der Funktion „ContourTracing()“ als Übergabeparameter. Ebenso wird die Anzahl der bis jetzt gefundenen Segmente um eins erhöht und übergeben, da dieser Wert dem Label des neuen Segmentes entspricht. Nach der Markierung der Kontur wird der Suchlauf fortgesetzt und die aktuelle Labelzahl auf den Wert 0 gesetzt. Die Anzahl der Segmente wird nicht bei jedem Aufruf der Funktion „ContourTracing()“ erhöht, da ein Aufruf auch bei einer inneren Kontur erfolgt. Eine äußere Kontur ist durch ein unmarkiertes Element, in der Segmentmatrix, mit dem Wert 1 in der Binärmatrix und der aktuellen Labelzahl gleich 0 gekennzeichnet, wobei eine innere Kontur durch ein unmarkiertes Element, in der Segmentmatrix, mit dem Wert 0 in der Binärmatrix und einer Labelzahl ungleich 0 gekennzeichnet ist.
Wird ein schwarzes, markiertes Matrixelement gefunden, das durch eine 1 in der Binärmatrix und einer Zahl ungleich 0 in der Segmentmatrix entspricht, so handelt es sich um einen Teil der Kontur. Damit alle Elemente dieser Zeile, die innerhalb der Kontur liegen, markiert werden können, bekommt die aktuelle Labelzahl den Wert der Kontur zugewiesen. Dadurch werden alle nachfolgenden Elemente mit dem Wert 1 in der Binärmatrix, in der Segmentmatrix mit dem Label des Segmentes markiert. Wird das Segment wieder verlassen, wird die aktuelle Labelzahl auf den Wert 0 gesetzt.
Bei einem Binärelement mit dem Wert 0, was einem weißen Bildpunkt entspricht, wird zuerst überprüft, ob die aktuelle Labelzahl 0 ist, oder nicht. Falls sie ungleich 0 ist, wird abgefragt, ob der Punkt in der Segmentmatrix bereits markiert wurde. Falls ja, dann handelt es sich um

den äußeren Rand eines Segmentes, da die Funktion „Tracer()“ die weißen Punkte um die Segmentkontur mit dem Wert -1 markiert, anderenfalls handelt es sich um den Randpunkt einer inneren Kontur. Jetzt erfolgt wiederum der Aufruf der Funktion „ContourTracing()“. Als Übergabewert dienen die aktuelle Labelnummer, sowie der Spaltenindex und der um eins verringerte Zeilenindex des Matrixelements, da dies ein Punkt der inneren Kontur ist und die Konturverfolgung von einem Element mit dem Wert 1 ausgeht. Für den Suchindex wird der Wert 1 übergeben. Danach folgt das Markieren der inneren Kontur sowie des weißen Randes in der Segmentmatrix. Anschließend wird die aktuelle Labelzahl wieder auf den Wert 0 gesetzt.
Wenn die Binärmatrix vollständig durchlaufen wurde, dann wird die Segmentmatrix elementweise kopiert. Die dadurch entstehende Ausgabematrix dient als Rückgabewert dieser Funktion. Dieser Schritt ist notwendig, denn so kann die nächste Binärmatrix segmentiert werden und gleichzeitig die segmentierte Ausgabematrix zur Gewinnung der Segmenteigenschaften genutzt werden.
Zum Speichern der Resultate der Segmentierung, muss ein Dateiname übergeben werden, um die Funktion „BildErstellen()“ aufzurufen.

5.2.4. BildErstellen()

Für die Darstellung der Resultate des Segmentierungsvorganges muss zuerst ein Bild in der Größe der Segmentmatrix erzeugt werden. Um die Segmente voneinander unterscheiden zu können, werden diese farblich markiert. Die Zuordnung der Farben erfolgt mittels der unterschiedlichen Markierungen der Segmente. Den ersten 10 Segmenten werden unterschiedliche Farben- oder Grauwerte zugeordnet. Alle Weiteren werden mit derselben Farbe abgebildet. Des Weiteren ist es auch möglich, den Rand sichtbar zu machen, indem man allen Elementen mit dem Wert -1 einen eigenen Farbwert zuordnet. Das Bild wird unter dem angegebenen Pfad gespeichert und vor der Extension _l eingefügt, um zu signalisieren, dass dieses Bild die Label der Segmente darstellt.

5.3. Segmentmerkmale

Bei dieser Anwendung werden einfache, annähernd runde Objekte verwendet. Die Klassifizierung erfolgt über die Fläche des jeweiligen Segments. Ein anderes wichtiges Merkmal ist die Lage der Segmente im Bild, welche durch den Schwerpunkt bestimmt werden kann. Der Segmentierungsalgorithmus liefert die Anzahl der gefundenen Segmente. Dadurch ist es möglich eine Matrix zu erstellen, in der die Merkmale aller Segmente gespeichert werden.
Für die Ermittlung der Merkmale wurden zwei Funktionen implementiert:

1. „GetLabelFeatures()“ sammelt für alle detektierten Segmente die benötigten Informationen
2. „GetFilteredLabelFeatures()“ löscht zu große beziehungsweise zu kleine Segmente aus der Matrix, da die Klassifizierung über die Fläche erfolgt.

Die Matrix mit den Segmenteigenschaften hat die Form der Tabelle aus Abbildung 49. Jede Zeile steht für ein Segmentlabel, weshalb dieses nicht zusätzlich gespeichert werden muss. Außerdem wird diese Information nicht mehr benötigt. Der grau hinterlegte Bereich beschreibt die Matrix der Segmenteigenschaften und beinhaltet den Flächeninhalt, gemessen als Summe der Bildpunkte oder Matrixelemente des Segments sowie die Summe der x- und y-Koordinaten beziehungsweise der Zeilen- und Spaltenindizes aller Segmentelemente, die zur Berechnung des Schwerpunktes benötigt werden. Die letzte Spalte enthält die Koordinaten des Schwerpunktes.

Segmentlabel	Fläche	Summe der x-Koordinaten	Summe der y-Koordinaten	x-Koordinate des Schwerpunktes	y-Koordinate des Schwerpunktes
1	[0,0]	[0,1]	[0,2]	[0,3]	[0,4]
…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…
n	[n-1,0]	[n-1,1]	[n-1,2]	[n-1,3]	[n-1,4]

Abbildung 49: Matrix der Segmenteigenschaften

Der Schwerpunkt ist für den weiteren Programmablauf besonders wichtig, da er die Lage der Objekte im Bild wiedergibt, welche für die Berechnung des Verschiebungsvektors und die Objektverfolgung benötigt wird.

5.3.1. GetLabelFeatures()

Diese Funktion benötigt die Segmentmatrix, sowie die Matrix in die sie die Segmenteigenschaften einträgt. Dazu wird die Segmentmatrix nach Labeln durchsucht. Wird ein Label gefunden, so wird in der entsprechenden Zeile, deren Nummer um eins kleiner ist als das Label, die Fläche um 1 erhöht, sowie die Summe der x- und y-Koordinaten um den jeweiligen Koordinatenwert erhöht.
Wenn die Matrix vollständig durchlaufen wurde, dann werden noch die Koordinaten der Schwerpunkte der Segmente berechnet. Dazu wird für die x-Koordinate die Summe aller x-Koordinaten durch die Fläche dividiert und genauso für die y-Koordinate verfahren.

5.3.2. GetFilteredLabelFeatures()

Diese Funktion benötigt neben den Übergabeparametern von „GetLabelFeatures()“ noch die minimale und die maximale Fläche für die Filterung der Segmente. Diese Filterung entspricht einer Klassifizierung der Objekte. Diese Klassifizierung erfolgt über zwei Merkmale:

1. Grauwert
2. Objektgröße.

Dabei handelt es sich um eine Klassifizierung nach der Quadermethode, die in Kapitel 2.5.2 beschrieben wurde.
Zuerst wird die Funktion „GetLabelFeatures()“ aufgerufen um die Segmenteigenschaften zu erhalten. Danach erfolgt eine Überprüfung der Fläche aller Segmente. Liegt diese außerhalb der Schranken so wird die Zeile gelöscht und alle anderen Zeilen werden um einen Index nach oben verschoben. Ebenso wird die Anzahl der Segmente verringert. Dadurch wird erreicht, dass nach der Überprüfung nur noch Segmente mit den richtigen Eigenschaften in der Liste stehen.

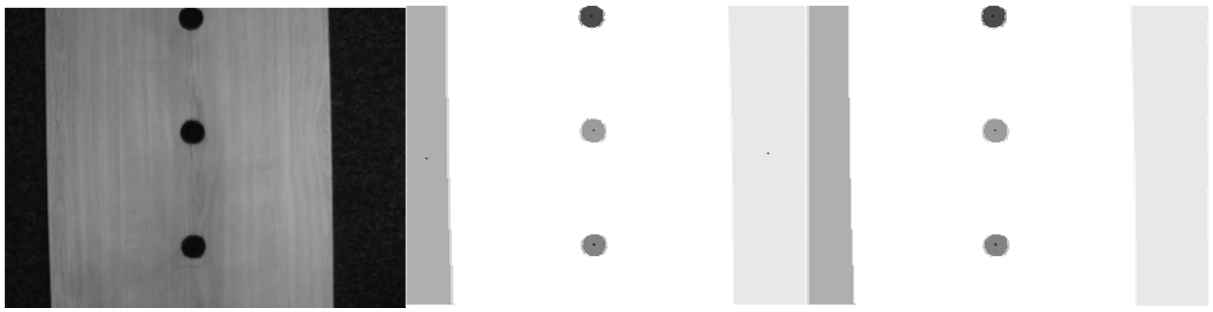


Abbildung 50: links Grauwertbild, Mitte Segmente, rechts Segmente gefiltert

Diese Funktion wird auch dazu genutzt, um in dem Segmentbild die gültigen Objekte zu markieren. Abbildung 50 zeigt links das Ausgangsbild. In der Mitte befindet sich ein Bild in dem alle Segmente markiert wurden, was durch das Einzeichnen des Schwerpunkts erfolgt. Da die beiden Streifen links und rechts im Bild Teil des Hintergrundes sind und nicht benötigt werden, können sie herausgefiltert werden. In Abbildung 51 sind die Segmenteigenschaften vor und nach der Filterung zu sehen.

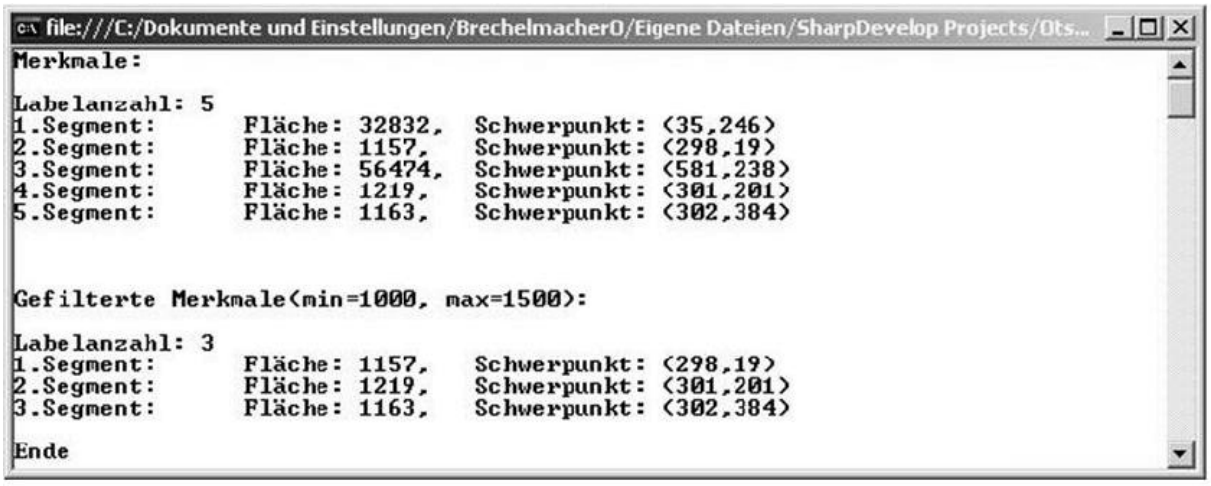


Abbildung 51: Konsole mit Segmenteigenschaften

Nach der Klassifizierung werden von jedem Bild nur noch die Schwerpunkte der gefilterten Segmente benötigt. Somit sind nur noch diese Segmente in der Objektliste vorhanden, welche in Abbildung 50 rechts eingezeichnet wurden.

5.4. Registrierung

Um dieselben Objekte aus zwei Bildern, die zu unterschiedlichen Zeitpunkten oder aus unterschiedlichen Perspektiven aufgenommen wurden, einander zuordnen zu können, benötigt man eine Registrierung. Das Ergebnis ist im einfachsten Fall ein Verschiebungsvektor. Dadurch ist es möglich, nach der Wahl eines Referenzbildes die Verschiebung der Objekte relativ zu den gleichen Objekten im Modellbild zu bestimmen.
Durch das Verschieben des Demonstrators werden Bilder aufgenommen, wobei dieselben Objekte in zwei aufeinander folgenden Bildern unterschiedliche Positionen aufweisen. Die Veränderung der Positionen kann mittels eines Vektors beschrieben werden. Zusätzlich kommt hinzu, dass Objekte den Bildbereich der Kamera verlassen und andere darin erscheinen. Da die Servomotoren nicht im Bildbereich oder direkt im Anschluss an den Bildbereich montiert sind, müssen die Positionen von den Objekten, die den Bildbereich verlassen, gespeichert und für jedes neue Bild aktualisiert werden. Dafür werden die Objektpositionen in einer Liste gespeichert und um den neu berechneten Verschiebungsvektor verschoben.
Für die Überprüfung der Registrierung können auch hier die Resultate gespeichert werden. Von besonderem Interesse sind die Positionen der Objekte außerhalb des Bildbereichs, die ebenfalls in das Ergebnisbild eingezeichnet werden. Damit der Vorgang der Bilderzeugung nicht zu lange dauert, wird auf die Abbildung des Hintergrundes verzichtet.
Mittels der Registrierung wird auch berechnet, wann die Motoren dem Objekt ausweichen müssen und auf welcher Position sich das Objekt befindet.

5.4.1. Verschiebungsvektor()

Für die Berechnung des Verschiebungsvektors benötigt die Funktion die Objekteigenschaften zweier aufeinanderfolgender Bilder. Beim Aufruf werden deshalb zwei Felder mit den im vorigen Abschnitt beschrieben gefilterten Objekteigenschaften übergeben, wobei die Ermittlung der Objekteigenschaften für jedes Bild nur einmal durchgeführt wird. Des Weitern wird noch die Anzahl der Objekte pro Bild benötigt. Ebenso werden noch die Objektliste, worin die Objekte außerhalb des Bildbereichs gespeichert werden, die Objektgröße und ein Feld mit dem Namen Motorposition, welches Parametern für die Motorsteuerung enthält, übergeben. Abbildung 52 zeigt die Aufteilung der Bereiche der Registrierung. Der Bildbereich der Kamera liefert die Daten für die Berechnung des Verschiebungsvektors. Verlassen Objekte diesen Bereich so werden diese in die Objektliste geschrieben und mit jedem neuen Bild weitergeschoben, bis sie den Zeichenbereich erreichen. Der Zeichenbereich beginnt bei den Motorachsen und ist so lang wie die Länge der Ausleger, an welchen die Stifte montiert sind. Mit Erreichen des Zeichenbereichs werden die Objekte nicht einfach gelöscht sondern so lange in der Liste gehalten, bis sie theoretisch gezeichnet worden sind (was später noch ausführlich beschrieben wird). Die nächsten drei Parameter für die Registrierung sind der „Zeichenrand“, welcher die Grenze zwischen dem Bereich Objektliste und Zeichenbereich verschieben kann, sowie „VorHalten“ und „AuslenkungHalten“, die für den Ausweichvorgang benötigt werden.
Mit dem Wert „VorHalten“ wird festgelegt, wie viele Bildpunkte vor oder nach dem berechneten Startzeitpunkt der Ausweichvorgang eingeleitet wird. Dies geschieht durch das Senden der Objektposition (genauer der x-Koordinate) an das aufrufende Programm, welches die Position der Motoren ändert.
Der Parameter „AuslenkungHalten“ bestimmt, wie lange der Ausweichvorgang dauert und wird als Vielfaches der Objektgröße angegeben.

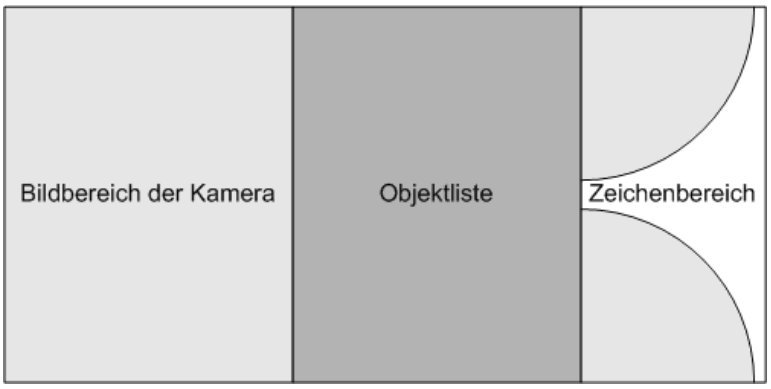


Abbildung 52: Bereiche der Registrierung

Der erste Schritt bei der Berechnung des Verschiebungsvektors ist die Zuordnung der Objekte des neueren Bildes zu den Objekten des älteren Bildes. Nach der Filterung der Grauwerte und der Objektgröße, wird unter der Annahme, dass der Abstand zwischen zwei Objekten eines Bildes wesentlich größer ist als der Betrag der Verschiebung zwischen den Bildern, der kürzeste Abstand zwischen den Objekten des ersten und des zweiten Bildes als Klassifizierungsmerkmal verwendet. Es ist nicht möglich die Objekte der einen Objektliste den Objekten mit demselben Index der anderen Objektliste zuzuordnen, da diese nicht immer übereinstimmen. Dies ist dann der Fall, wenn ein Objekt den Bildbereich verlässt, beziehungsweise wenn ein neues Objekt in den Bildbereich kommt, oder aber beides der Fall ist. Für die Berechnung wird eine Matrix verwendet, deren Elemente die euklidischen Abstände der Objekte des älteren Bildes zu den Objekten des neueren Bildes sind. Die

Zeilenindizes stehen für die Objektnummern des alten Bildes, die Spaltenindizes für die Objektnummern der neueren Aufnahme. In Abbildung 53 ist die Abstandsberechung für das Objekt 2 der linken Aufnahme, welche als Referenzbild für die Registrierung benutzt wird. Die rechte Aufnahme ist gegenüber der linken in Richtung der Pfeile verschoben. Für diesen Fall würden die Matrixelemente auf der Hauptdiagonale die kürzesten Anstände darstellen. Würden hingegen in der linken Aufnahme nur zwei Objekte vorhanden sein, dann wäre dieses Kriterium unmöglich.

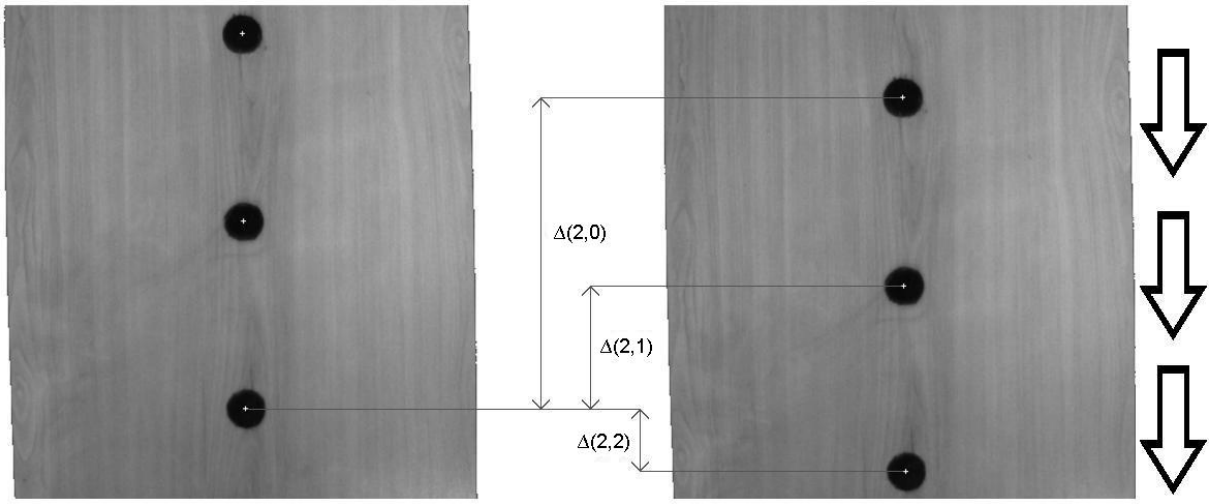


Abbildung 53: Berechnung der Objektabstände in y-Richtung

Aus diesem Grund wird zuerst ein Vektor aus der Differenz der x- und y-Koordinaten Δx und Δy berechnet, wie in Abbildung 53 dargestellt, und anschließend der Betrag des Vektors in die Abstandsmatrix geschrieben. Abbildung 54 zeigt die Berechnung des Verschiebungsvektors für zwei aufeinander folgende Bilder. Unter der Nummer der Bilder wird die Anzahl der ungefilterten und gefilterten Segmente angezeigt. In der Mitte ist die Abstandsmatrix dargestellt. Die Zelle (1,0) enthält den euklidischen Abstand vom dem mittleren Objekt des Referenzbildes zu dem obersten Objekt des Modellbildes aus Abbildung 53. Als nächster Schritt wird mit der Funktion „GetMinZeilenelement()“ aufgerufen, welche das kleinste Element einer Matrixzeile ermittelt.

```
1.Bild: 32      2.Bild: 33

Label Bild 1: 5          gefiltert: 3
Label Bild 2: 5          gefiltert: 3

Registrierung

Abstände:
(0,0):62  ¦ (0,1):245 ¦ (0,2):425 ¦
(1,0):120 ¦ (1,1):63  ¦ (1,2):243 ¦
(2,0):302 ¦ (2,1):119 ¦ (2,2):61  ¦

Kürzester Abstand:
Element(0,0): 62
Element(1,1): 63
Element(2,2): 61

Häufigster Wert: 61 (1x)

Verschiebungsvektor:(-1, -62)
```

Abbildung 54: Ermittlung des Verschiebungsvektors, 3x3 Matrix

Somit stehen für alle Zeilen die kürzesten Abstände zur Verfügung. Diese werden in einem Feld gespeichert, wobei dessen Zeilenindex dem Zeilenindex der Matrix entspricht, was wiederum für die Objektnummer aus dem Referenzbild steht. Um den Abstand wieder einem Objekt aus dem Modellbild zuordnen zu können, wird dessen Index ebenfalls in dem neuen Feld gespeichert. Für eine unsymmetrische Abstandsmatrix können die Abstände sehr stark variieren. Wird im Modellbild das unterste Objekt nicht mehr angezeigt, so würde der kürzeste Abstand etwa dem Abstand zwischen zwei Objekten desselben Bildes entsprechen. Dadurch kann nicht einfach der Mittelwert aller ermittelten Abstände berechnet werden, sondern es wird der häufigste Wert ermittelt. Dazu wird ein Feld mit zwei Spalten erzeugt, welches so viele Zeilen wie die Abstandsmatrix besitzt. In der ersten Spalte werden die ermittelten kürzesten Abstände eingetragen. Die zweite Spalte wird später zum Aufsummieren genutzt und deshalb vorerst mit dem Wert Eins initialisiert.
Anschließend wird dieses Feld zeilenweise nach aufsteigenden Abstandswerten mit der Funktion „BubblesortArray()“ sortiert. Da die Werte jetzt in geordneter Form vorliegen, wird vom ersten Element beginnend, der Wert mit allen nachfolgenden verglichen. Bei einer Übereinstimmung, wird die zweite Spalte um eins erhöht und in die Zellen des Elements mit dem gleichen Wert, aber einem höheren Index, Null geschrieben, damit sie für weitere Vergleichsoperationen nicht genutzt werden kann. Ist dieser Vorgang abgeschlossen, so wird das Feld durchsucht und überprüft, welcher Wert am Häufigsten aufgetreten ist. Sind mehrere Werte gleich oft aufgetreten, dann wird der kleinere Abstand als Referenz genommen.

```
1.Bild: 1001    2.Bild: 1002

Label Bild 1: 8          gefiltert: 6
Label Bild 2: 8          gefiltert: 6

Registrierung

Abstände:
(0,0):19  | (0,1):112 | (0,2):200 | (0,3):293 | (0,4):383 | (0,5):471 |
(1,0):70  | (1,1):23  | (1,2):111 | (1,3):204 | (1,4):294 | (1,5):382 |
(2,0):158 | (2,1):65  | (2,2):23  | (2,3):116 | (2,4):206 | (2,5):294 |
(3,0):250 | (3,1):157 | (3,2):69  | (3,3):24  | (3,4):114 | (3,5):202 |
(4,0):341 | (4,1):248 | (4,2):160 | (4,3):67  | (4,4):23  | (4,5):111 |
(5,0):432 | (5,1):339 | (5,2):251 | (5,3):158 | (5,4):68  | (5,5):20  |

Kürzester Abstand:
Element(0,0): 19
Element(1,1): 23
Element(2,2): 23
Element(3,3): 24
Element(4,4): 23
Element(5,5): 20

Häufigster Wert: 23 (3x)

Verschiebungsvektor:(0, -22)
```

Abbildung 55: Ermittlung des Verschiebungsvektors, 6x6 Matrix

Abbildung 55 zeigt die Abstandsmatrix für jeweils 6 Objekte pro Bild, sowie die Liste der kürzesten Abstände, wobei der Wert 23 dreimal vertreten ist. Für die Berechnung des Verschiebungsvektors werden alle Vektoren berücksichtigt, deren Betrag im Bereich von fünf Bildpunkten um den häufigsten Wert liegt. Damit wird verhindert, dass Vektoren in die Berechnung einfließen die deutlich größer als die eigentliche Verschiebung sind. Solche Vektoren treten auf, wenn ein Objekt aus dem Bildbereich verschwindet, beziehungsweise eintritt. Zusätzlich wird überprüft, ob ein Objekt zweimal in der Liste auftritt. Dies ist bei einer unsymmetrischen Abstandsmatrix der Fall. Verlässt beispielsweise im Modellbild ein Objekt den Bildbereich, so ist die Anzahl der Zeilen größer als die Anzahl der Spalten. Dadurch wird in der Liste der kürzesten Anstände zweimal der Abstand zu ein und demselben

Objekt des Modellbildes ausgegeben. Tritt dieser Fall auf, so wird nur der kürzere Abstand berücksichtigt, falls beide innerhalb des gültigen Wertebereichs liegen. Zusätzlich wird die Anzahl der gültigen Objekte bestimmt. Liegt der Wert aus der Abstandsliste innerhalb der Schranken, so werden mittels des Zeilenindex und dem gespeichert Index die Schwerpunktskoordinaten der jeweiligen Objekte ermittelt.
Die Differenz der x- und y-Koordinaten derselben Objekte aus den beiden Bildern beschreibt die Komponenten des Verschiebungsvektors zwischen den Bildern. Diese Berechnung erfolgt für alle gültigen Elemente der Abstandsmatrix, wobei die Komponenten des Verschiebungsvektors als Summe der einzelnen Objektverschiebungen berechnet werden. Zuletzt werden die x- und die y-Komponente des Verschiebungsvektors noch durch die Anzahl der gültigen Objektpaarungen dividiert, was einer Mittelung über alle Verschiebungen entspricht.
In den Abbildung 54 und Abbildung 55 ist in der untersten Zeile der Verschiebungsvektor für den jeweiligen Datensatz zu sehen.
Für die Berechnung der neuen Objektpositionen werden die Objekte des Referenzbildes um den Verschiebungsvektor verschoben. Eigentlich könnten auch die Objektpositionen des neueren Bildes benutzt werden, da diese bereits bekannt sind. Doch für den Fall, dass ein Objekt den Bildbereich der Kamera verlassen hat, könnte man dessen Position nicht mehr bestimmen. Wird bei der Berechnung der neuen Position festgestellt, dass ein Objekt über den Bildrand hinaus verschoben wurde, so werden dessen Koordinaten in einer Liste gespeichert. Diese Liste trägt den Namen Objektliste und enthält die Koordinaten aller Objekte, die den Sichtbereich der Kamera verlassen haben und noch nicht von den Auslegern umfahren wurden. Bei der Objektliste handelt es sich um eine zyklische Liste, deren erste Zelle den nächsten freien Index enthält. Nachdem die Berechnung des Verschiebungsvektors erfolgt ist, werden die Positionen aller Objekte in der Liste neu berechnet.

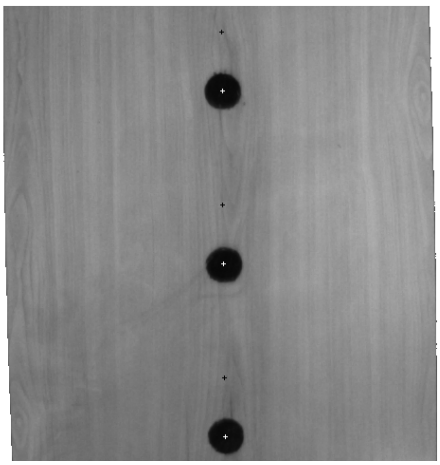


Abbildung 56: Berechnung der Objektkoordinaten (schwarze Kreuze: Objekte des Referenzbildes, weiße Kreuze: berechnete Positionen des aktuellen Bildes)

In Abbildung 56 ist die Berechnung der aktuellen Objektkoordinaten dargestellt. Die schwarzen Kreuze sind die Objektkoordinaten des Referenzbildes, die im nachfolgenden Bild eingezeichnet wurden. Von diesen wurde der Verschiebungsvektor subtrahiert und mit einem weißen Kreuze markiert. Die berechneten Koordinaten stimmen sehr gut mit den Objektschwerpunkten des aktuellen Bildes überein.
Wenn die untere Grenze für die Filterung der Segmentfläche sehr niedrig eingestellt wird, dann werden auch kleine Objekte erkannt, die eigentlich nur mehr einem Teil des tatsächlichen Objektes entsprechen. Dies kann dazu führen, dass in drei aufeinanderfolgenden Bildern zwei Einträge für dasselbe Objekt in die Objektliste erfolgen. Befindet sich im ersten

Bild das Objekt am unteren Rand und im zweiten Bild ist auch noch ein Stück zu erkennen, so erfolgt ein Eintrag in die Objektliste, falls der berechnete Objektschwerpunkt nicht mehr im Bildbereich liegt. Ebenso erfolgt ein Eintrag in die Objektliste, falls der Teil des Objektes im zweiten Bild noch als Objekt erkannt wird. Der Schwerpunkt der Teilfläche liegt etwas höher als der eigentliche Objektschwerpunkt und deshalb erfolgen zwei Einträge für ein Objekt. Da diese sehr knapp beieinander liegen, werden die Motoren deswegen lediglich etwas länger auf der Ausweichposition gehalten.
Wenn die Ausleger nicht gerade ein Objekt umschreiben, dann befinden sie sich in der Bildmitte, beziehungsweise bewegen sie sich zwischen der Bildmitte und dem seitlichen Bildrand hin und her. Das Einleiten des Ausweichvorganges kann nicht an der Grenze zu dem Zeichenbereich erfolgen, denn für Objekte in der Bildmitte wäre das zu spät und für Objekte weiter außen zu früh. Dies würde einen extrem langen Ausweichvorgang nach sich ziehen. Deshalb wird das Starten der Ausweichbewegung abhängig von der Verschiebung und der Objektposition in x-Richtung bestimmt. Zusätzlich kann mit dem Parameter „VorHalten“ eingestellt werden, ob die Bewegung früher oder später eingeleitet werden soll. Der Ausweichvorgang wird gestartet, sobald die y-Koordinate des Objektmittelpunktes den Wert „MotorAusrichtungStart“ erreicht hat. Dieser setzt sich aus der Bildhöhe b, dem Abstand von der Bildunterkante bis zu den Motorachsen m, der y-Komponente des Verschiebungsvektors v_y und eines Parameters p_x aus dem Feld Motorposition welcher die x-Koordinate des Objektes berücksichtigt (worauf später noch eingegangen wird) wie folgt zusammen:

$$MotorAusrichtungStart = b + m - v_y + p_x . \quad \textbf{(5.2)}$$

Die Variable „MotorAusrichtungStop“ gibt den Wert an, an dem der Ausweichvorgang beendet wird. Diese Schranke besteht aus dem Wert von „MotorAusrichtungStart“, sowie einem Vielfachen des Objektdurchmessers, einstellbar über den Parameter „AuslenkungHalten“ und einer x-abhängigen Komponente aus dem Feld Motorposition.
Die Funktion „Verschiebungsvektor()“ hat als Rückgabewert die x-Koordinate der Objektposition, solange sich das Objekt zwischen den Schranken des Ausweichvorganges befindet. Sobald dieser vorbei ist und keinem Objekt ausgewichen werden muss, wird ein Wert, größer als die Bildlänge, zurückgeliefert. Da diese Position eigentlich nicht auftreten kann, wird dem Hauptprogramm dadurch vermittelt, dass die Ausleger wieder in die Bildmitte steuern können.
Eine Besonderheit der Registrierung ist, dass die Bilder auch besonders schnell gespeichert werden können. Es werden lediglich die Objektschwerpunkte im Bildbereich, die Objektschwerpunkte der Elemente der Objektliste, das Ende des Bildbereichs der Kamera, sowie die Schwellen für den Ausweichvorgang eingezeichnet. Auf eine Darstellung des Hintergrundes des Kamerabildes wird verzichtet. Dies ermöglicht ein schnelles Erstellen der Bilder, da nur einige Punkte eingezeichnet werden müssen. Dadurch ist es möglich auch während des Betriebes die Resultate abzuspeichern, was für eine Fehlersuche sehr nützlich ist, ohne die Bildrate stark zu beeinflussen.
Abbildung 57 zeigt die Positionsberechnung eines Objektes von der Detektion im Bildbereich der Kamera (links oben) bis zum Ausweichvorgang (rechts unten). Dabei werden der Bildbereich der Kamera, die Objektliste und der Zeichenbereich durch Linien getrennt. Das Bild in der Mitte oben zeigt die erste Berechnung der Objektposition außerhalb des sichtbaren Bereiches. Anschließend werden die Positionen immer weiter berechnet bis der Ausweichvorgang eingeleitet wird. Dies ist im Bild unten in der Mitte zu sehen. Die beiden grauen Linien sind die Grenzen des Ausweichvorganges. Diese sind selbst für ein Objekt nicht starr, da bei dem Umfahren des Objektes auf seitliche Verschiebungen des Demonstrators geachtet werden muss.
Im Bild links unten sieht man wie aus einem Objekt im sichtbaren Bereich ein Eintrag in die Objektliste erfolgt.

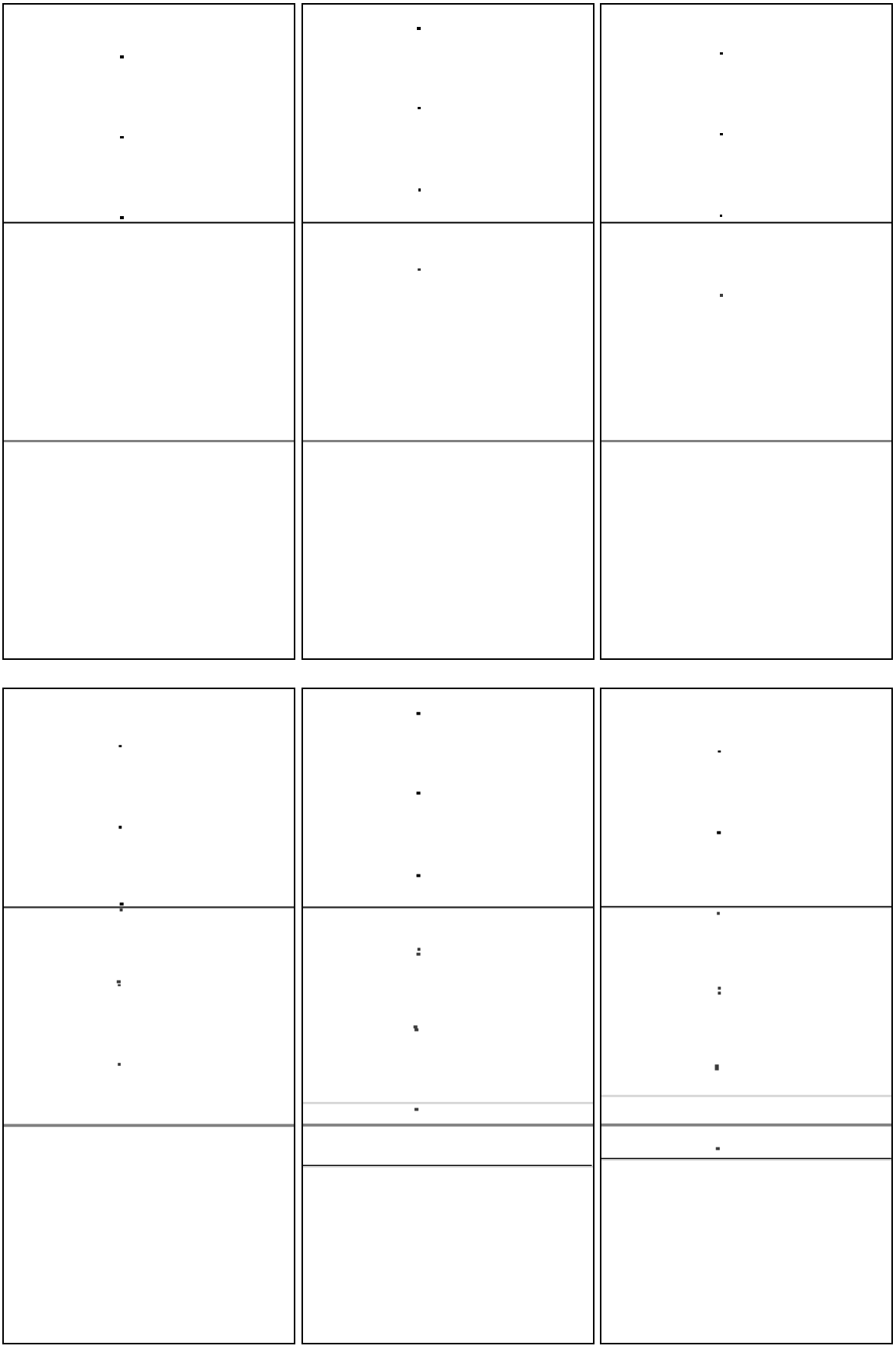


Abbildung 57: Ergebnisbilder der Positionsberechnung

Dadurch entstehen in der Objektliste zwei Einträge für dasselbe Objekt.
Wie oben beschrieben, wird in diesem Fall in beiden Bildern das Objekt erkannt. Da im nachfolgenden Bild der Schwerpunkt aufgrund der kleineren Fläche des Objekts verschoben ist, liegt die errechnete Position höher als der Punkt in der Objektliste. Der Eintrag in die Objektliste erfolgt wenn der Objektschwerpunkt innerhalb von 10 Bildpunkten über der Unterkante des sichtbaren Bereiches zu liegen kommt. Dadurch wird verhindert, dass Objekte, deren errechnete Position gerade noch im sichtbaren Bereich ist und somit nicht in die Objektliste eingetragen werden, verlorengehen, da sie bei dem nächsten Registrierungsvorgang in keinem der beiden Bilder zu sehen sind.
Die Ergebnisbilder der Registrierung werden gespeichert, wenn ein Pfad übergeben wird.
Damit die Bilder der Registrierung zugeordnet werden können, wird vor der Extension _r in den Dateinamen eingefügt.

5.4.2. Hilfsfunktionen

Für die Berechnung des Verschiebungsvektors wurden für das Arbeiten mit Listen und Matrizen einige nützliche Funktionen implementiert.

GetMinZeilenelement()

Diese Funktion bekommt eine Matrix übergeben und ermittelt für jede Zeile den kleinsten Wert. Dazu wird ein Vergleichsoperator benötigt, welcher am Anfang jeder neuen Zeile auf den Wert der Bildlänge gesetzt wird, da dies der maximalen x-Koordinate entspricht. Danach wird dieser Wert mit dem Element in der ersten Spalte verglichen, ist es kleiner, so wird dessen Wert in den Vergleichsoperator geschrieben. Diese Operation wird für jede Spalte durchgeführt und liefert so den kleinsten Wert jeder Zeile. Die Funktion liefert ein Feld zurück, wobei die kleinsten Elemente pro Zeile und die zugehörigen Indizes des Elements der Ursprungsmatrix eingetragen werden.

BubblesortArray()

Diese Funktion dient zum Sortieren einer Matrix. Es werden nur die Elemente in der ersten Spalte aufsteigend sortiert und die zugehörigen Daten der jeweiligen Zeile wechseln ihre Position mit.
Das Sortieren erfolgt durch einen Vergleich zweier Nachbarn. Ist das erste Element kleiner als das zweite, dann befinden sie sich in der richtigen Reihenfolge. Ist das jedoch nicht der Fall, dann werden sie vertauscht. Dieser Vorgang wird so oft durchgeführt, bis die erste Spalte in sortierter Reihenfolge vorhanden ist. Hierbei kann es vorkommen, dass die Matrix mehrmals durchlaufen werden muss. Dieser Sortieralgorithmus ist nicht besonders effizient, aber er ist sehr einfach zu implementieren und da für diese Anwendung Matrizen mit weniger als 10 Zeilen verwendet werden durchaus geeignet.

5.5. Motorposition()

Diese Funktion dient zur Berechnung der Motorpositionen aus den Bildkoordinaten. Damit nicht für jedes Objekt die Koordinaten neu berechnet werden müssen werden diese bei dem Programmstart berechnet und in einen Feld gespeichert.
Für jede x-Koordinate im Bild steht eine Zeile der Matrix zur Verfügung, welche folgende Werte enthält:

1. Position Motor 1 (linker Motor)
2. Position Motor 2 (rechter Motor)
3. Verschiebung für das Starten des Motors
4. Dauer des Ausweichvorganges, wie viele Bildpunkte die Position gehalten werden soll

Da die Ausleger der Motoren die Bildmitte nicht überragen beziehungsweise ihren Arbeitsbereich nicht kreuzen, werden nicht immer beide Motoren für den Ausweichvorgang benötigt.
Das Speichern der Daten in einer Liste hat den Vorteil, dass während der Registrierung schon auf die Liste zugegriffen werden kann und die Position für das Starten des Motors nicht separat berechnet werden muss.
Für die Berechnung der Motorpositionen sind einige Parameter notwendig: Die Länge der Ausleger muss bekannt sein, wobei dieser Wert in Pixel benötigt wird. Ebenso ist die Breite des Bildes b sowie der Durchmesser oder die Breite der Objekte o notwendig. Da die Motoren in einem Bereich von 90° positionieren, wird der Parameterwert M für diese Position benötigt. Ebenso muss der Abstand der beiden Motorachsen zum jeweils näheren Bildrand bekannt sein.
Abbildung 58 zeigt die geometrischen Gegebenheiten für die Berechnung der Motorpositionen. Die Zeichnung wurde soweit vereinfacht, dass die Abstände der Motoren m von den Bildrändern gleich groß gewählt wurden. Wird die x-Koordinate eines Objektschwerpunktes bestimmt, so entspricht das der Position $m+\Delta x_1$. Die waagrechte Line entspricht der Grenze zum Zeichenbereich. Ab diesem y-Wert kann gezeichnet werden.
Die Ausleger, die an den Servomotoren befestigt sind, werden durch die Pfeile mit der Länge r dargestellt und können einen Viertelkreis abfahren. Der linke Motor wurde so montiert, dass der Ausleger bei Position 0 in Fahrtrichtung zeigt und bei Position 1024 quer zur Fahrtrichtung steht.

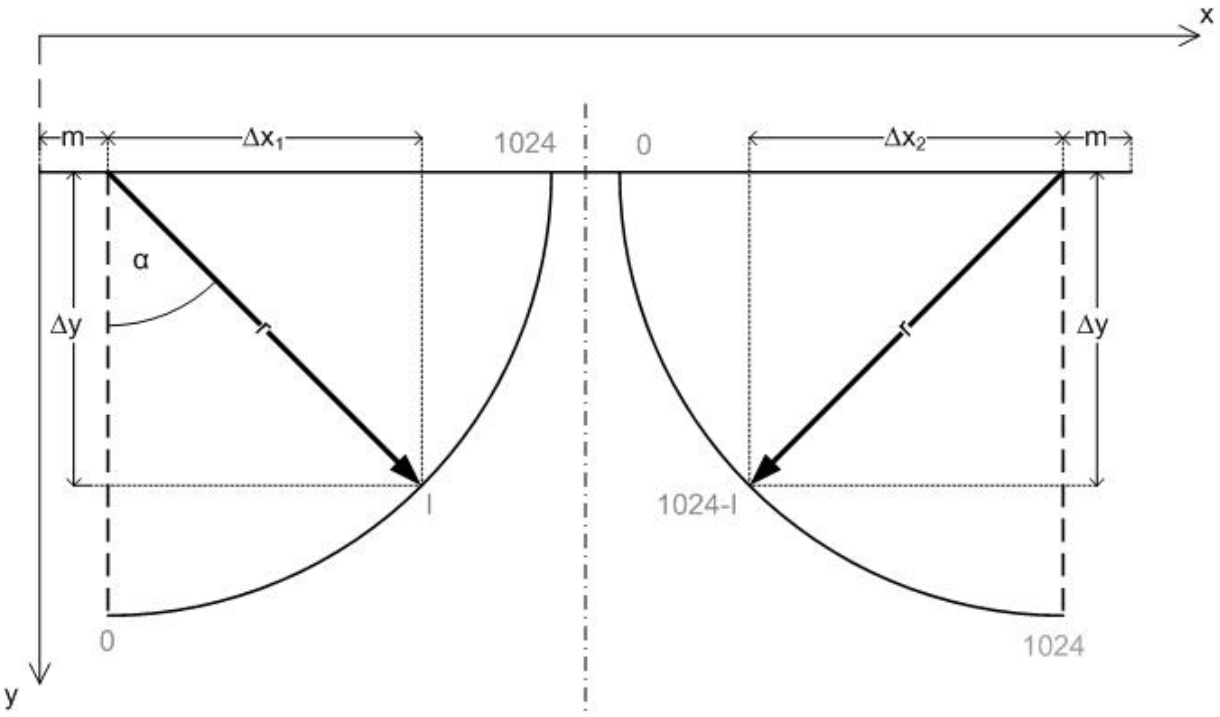


Abbildung 58: Bestimmung der Motorpositionen

Der rechte Motor ist gegenüber dem linken Motor um 90° im Uhrzeigersinn verdreht. Um nicht für beide Motoren die Positionen separat berechnen zu müssen, reicht es die Position I des linken Motors von 1024 (was der Position bei 90° entspricht), zu subtrahieren um die Position des rechten Motors zu erhalten für den Fall, dass Δx_1 gleich Δx_2 ist.
Dabei werden vier Bereiche berücksichtigt:

1. Das Objekt liegt außerhalb des Zeichenbereichs. Das heißt der Punkt liegt weniger als der Motorabstand plus der Objektbreite vom Bildrand entfernt. Für diesen Fall werden die Motoren in der Bildmitte positioniert.
2. Das Objekt liegt im Zeichenbereich des linken Motors. So enthält die Positionsliste den Wert für den Ausweichvorgang, der rechte Motor wird in der Bildmitte (Position 0) gehalten.
3. Liegt das Objekt im Zeichenbereich des rechten Motors, so weicht nur dieser aus und der linke Motor bleibt in der Mittelstellung (Position 1024).
4. Befindet sich das Objekt in der Bildmitte so sind beide Motoren am Ausweichvorgang beteiligt, welcher schon gestartet wird, bevor das Objekt die Grenze zum Zeichenbereich überschritten hat.

Für Bereich 1 werden für die Motoren die Position 1024 beziehungsweise 0 eingetragen, sowie die Werte für die Verschiebung des Startzeitpunktes und die Dauer des Positioniervorganges auf den Wert 0 gesetzt.
Die Berechnung der Motorpositionen für die Bereich 2 und 3 erfolgt in einem Schritt. Für die Berechnung der Position I, des linken Motors, wird die x-Koordinate des Bildpunktes benötigt. Von dieser wird der Abstand der Motorachse vom Bildrand m abgezogen und liefert somit das für die Winkelberechnung benötigte Δx_1. Da der Motor dann genau den Objektmittelpunkt ansteuern würde wird noch zusätzlich die Objektbreite o von der x-Koordinate subtrahiert, was aber in der Grafik nicht berücksichtigt wurde. Weiters werden noch die Länge der Ausleger und er Index I bei 90° benötigt.

$$\alpha = \arcsin(\frac{\Delta x_1}{r}) \qquad \textbf{(5.3)}$$

beschreibt den Winkel zwischen der Position 0 des linken Motors und der Zielposition des Auslegers. Anschließend wird der Winkel in Grad umgerechnet und mit dem Verhältnis Index pro Grad multipliziert und liefert somit die Position

$$I = \frac{\alpha * 180°}{\pi} * \frac{M}{90°} = \frac{\alpha * 2 * M}{\pi}. \qquad \textbf{(5.4)}$$

Die berechnete Position wird in der Positionsliste für die x-Koordinate und den Motor 1 eingetragen. Liegt der x-Wert außerhalb des Bereiches, welchen der rechte Motor anfahren kann, so wird er auf die Mittelposition 0 gesetzt. Die berechnete Position kann ebenso benutzt werden um einem Objekt auszuweichen, welches sich auf der Position Bildrand-m-Δx_2 befindet. Auf dem entsprechenden Index wird für den rechten Motor die Position 1024-I für den Ausweichvorgang und für den linken Motor die Position 1024 eingetragen.
Für den Startpunkt des Ausweichvorganges berechnet man den Wert Δy mit

$$\Delta y = r * \cos(\alpha). \qquad \textbf{(5.5)}$$

Dieser Wert gibt an wie weit hinter der Grenze zum Zeichenbereich das Objekt von den Auslegern umfahren werden kann.
Von diesem Wert wird noch die Objektbreite abgezogen, damit ein Abstand zum Objekt bleibt und das Ergebnis in die Liste geschrieben. Für den Endpunkt des Ausweichvorganges wird zu Δy noch die Objektbreite addiert. Dieser Wert reicht aus, da schon aufgrund der Abweichung der berechneten x-Koordinate vom tatsächlichen Objektschwerpunkt eine größere Ausholbewegung erwirkt wird, wodurch beim Zurückstellen des Motors auf die Mittelstellung etwas mehr Spielraum zum Objekt bleibt.
Für Objekte die nahe der Bildmitte liegen wird der Ausweichvorgang 2,5 Objektbreiten vor der Grenze zum Zeichenbereich eingeleitet und 2,5 Objektbreiten danach wieder beendet. Die Positionen werden weiterhin wie oben beschrieben berechnet. Der letzte berechnete Wert für

diesen Bereich (bei Δx_1 ist gleich der Läge des Auslegers) dient als Ausweichposition für den Ausweichvorgang in der Bildmitte.
Abbildung 59 zeigt den Ausweichvorgang für ein Objekt, das sich im linken Bildbereich befindet. Die Spitze des Auslegers stellt den Stift für das Umschreiben des Objektes dar. Dies ist der einzige Punkt der den Boden berührt, deshalb ist es möglich mit dem Ausleger über das Objekt hinweg zu fahren, ohne es zu berühren. In Bild 1 liegt das Objekt noch vor dem linken Ausleger, aber es wäre viel zu früh um den Ausweichvorgang einzuleiten, weil sehr große Ausweichbewegungen um das Objekt herum notwendig wären (da erst nach dem Objekt wieder auf die Mittelposition zurückgekehrt werden kann). Dies würde bei einer Reihe von Objekten mit derselben oder einer ähnlichen x-Koordinate dazu führen, dass der Ausleger nur noch neben den Objekten positioniert wird und die Mittelstellung nicht mehr einnehmen kann. Bild 3 zeigt, wie der Ausleger den Stift rechts am Objekt vorbeibewegt und eine Position einnimmt, welche, wie im Teilbild 4 zu erkennen ist, links neben dem Objekt liegt. Dadurch kann das Objekt passiert werden, ohne es zu berühren. Hat das Objekt den Zeichenbereich verlassen, so wird der Positioniervorgang beendet und der Ausleger kehrt in die Mittelposition zurück. Teilbild 6 zeigt die Bewegung der beiden Ausleger. Da das Objekt im Zeichenbereich des linken Auslegers liegt, bleibt der Ausleger des rechten Motors in der Mittelposition. Der Ausleger des linken Motors bewegt sich vor dem Objekt auf die berechnete Position, verharrt dort bis das Objekt vorbei ist und kehrt in die Mittellage zurück. Die tatsächliche Ausweichbewegung kommt diesem Modell sehr nahe.

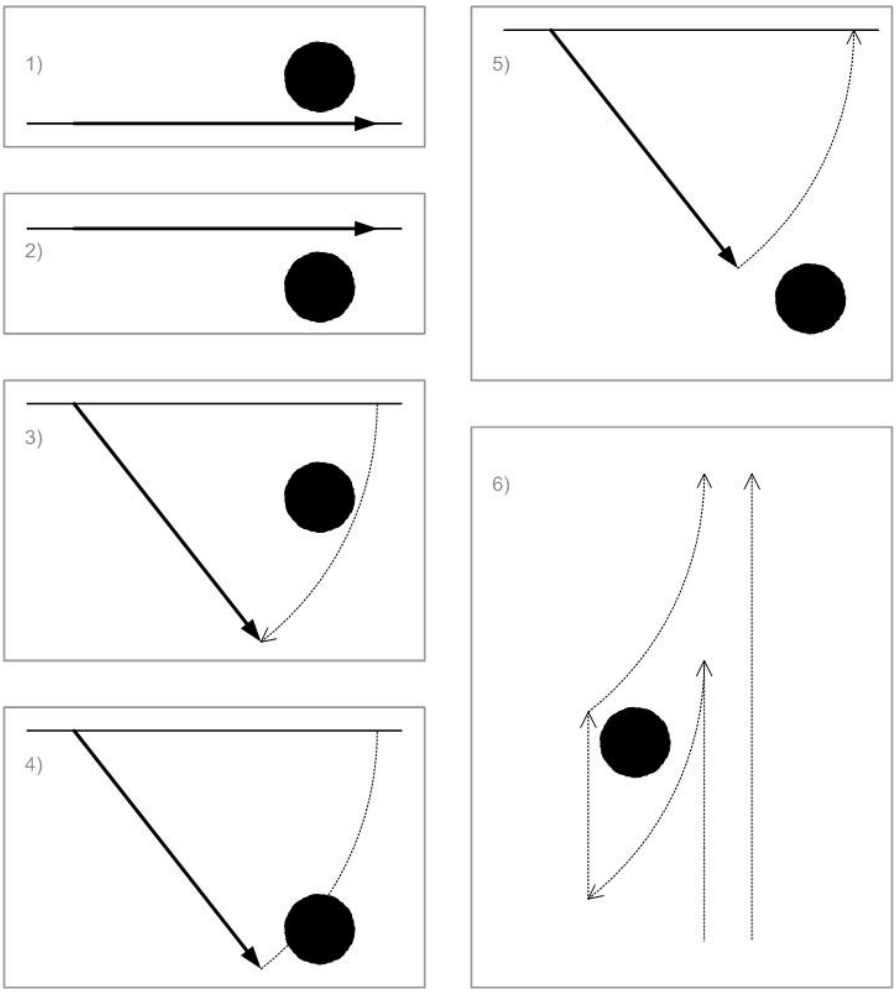


Abbildung 59: Ausweichvorgang des linken Motors

Befindet sich das Objekt ungefähr in der Bildmitte, so sind beide Servomotoren an dem Ausweichvorgang beteiligt. Für Objekte die in einem Bereich von der doppelten Objektbreite um die Bildmitte liegen, wurde eine spezielle Steuerung implementiert. Dazu wird der Index an der Grenze dieses Bereiches aus der bereits vorhandenen Liste ausgelesen und für den gesamten Bereich konstant gehalten. Der Ausweichvorgang wir konstant auf 2,5 Objektbreiten vor der Grenze zum Zeichenbereich eingeleitet und 2,5 Objektbreiten danach wieder beendet.
Abbildung 60 zeigt den Ausweichvorgang für ein Objekt in der Bildmitte. Der Ausweichvorgang startet schon vor dem Zeichenbereich, da beim Überschreiten der Grenze die Ausleger bereits in der Ausweichposition sein müssen (siehe Teilbild 2). Anschließend bleiben sie so lange in dieser Position, bis das Objekt vorbei ist. Danach kehren sie wieder in die Mittelposition zurück. Das Teilbild 5 stellt die Ausweichbewegung der Stifte um das Objekt dar. Liegen die Objekte sehr knapp hintereinander, so wird zwischen diesen nicht in die Mittelposition zurückgekehrt.

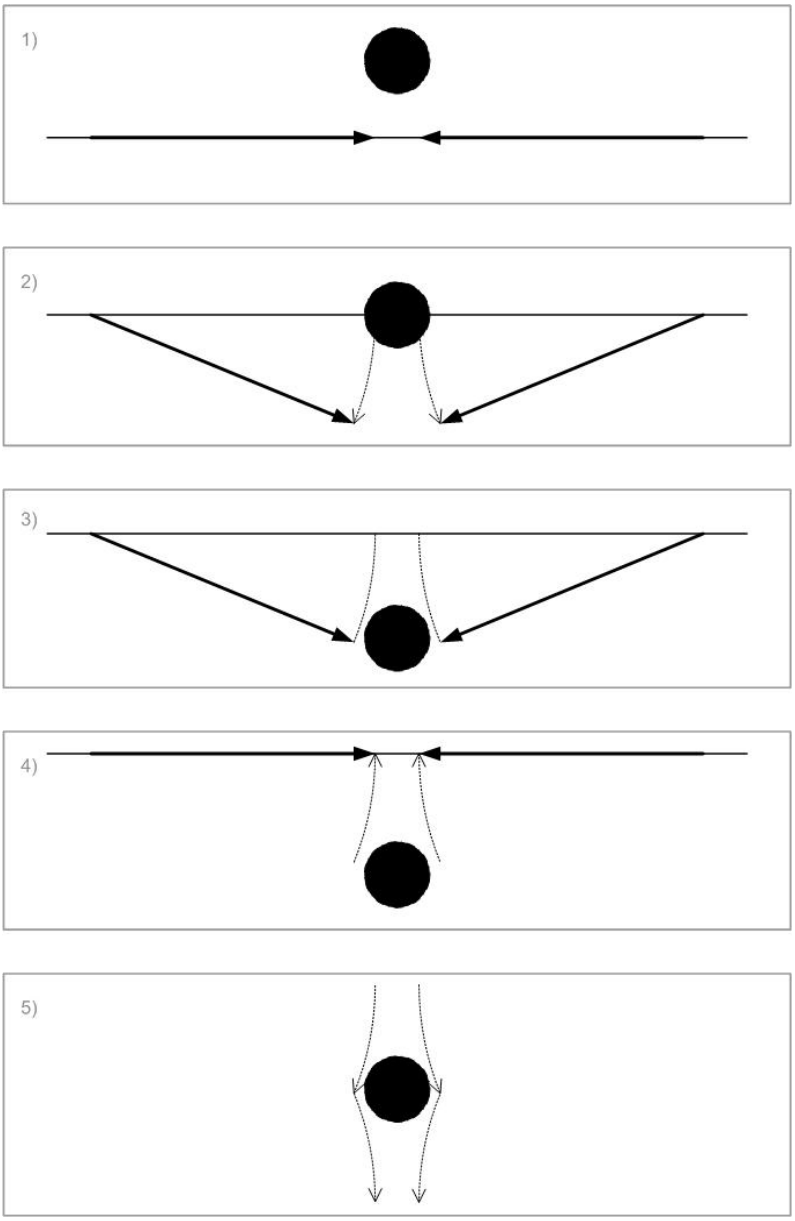


Abbildung 60: Ausweichvorgang für den mittleren Bereich

Die Dauer des Ausweichvorganges für den mittleren Bereich wurde mittels der festen Grenzen etwas großzügiger ausgelegt, damit für den Fall knapp hintereinander liegender Objekte jedem ausgewichen werden kann. Wird das Objekt nur sehr knapp umfahren, dann kann es passieren, dass das beim Abbremsen des Servomotors in der Mittelposition bereits ein Objekt berührt oder überfahren wird, da die neue Position noch nicht bei den Frequenzumrichtern angekommen ist. Ein weiteres Problem bei der Positionierung ist das Überschwingen der Ausleger, verursacht durch die hohen Beschleunigungswerte und die Masse der Ausleger. Der Ausleger kann bei einer höheren Positioniergeschwindigkeit nicht exakt an der Mittelposition abgebremst werden, sondern führt die Kreisbewegung in Richtung des neuen Objektes fort, bis der Regler des Frequenzumrichters ihn auf die exakte Position führt. In Abbildung 61 sind die Einträge aus der Positionsliste für den Ausweichvorgang bei einem Objekt mit 20 Bildpunkten Objektbreite. Der obere Bildteil mit dem weißen Hintergrund zeigt die Motorpositionen für die beiden Servomotoren. Im rechten Teilbild sind die Motorpositionen von beiden Motoren aus der Liste ausgelesen und in das Bild eingetragen worden. Dabei entspricht die obere Bildkante dem Index 0, welcher bis zum hellgrauen Bildbereich auf den Wert 1024 ansteigt. Die Bildbreite stellt die x-Koordinate dar. Für ein besseres Verständnis wurden im linken Teilbild die Position des rechten Motors von 1024 subtrahiert um den Verlauf der Ausweichbewegung besser nachvollziehen zu können. Dabei entspricht die schwarze Kurve dem linken und die graue Kurve dem rechten Servomotor. Anhand des rechten Bildes kann man gut erkennen, wo die einzelnen Steuerbereiche liegen. An den Rändern bleiben beide Motoren in der Bildmitte, was dem Index 0 beziehungsweise 1024 entspricht. Anschließend beginnt der Zeichenbereich des linken Motors, wobei der rechte Motor noch immer in Mittelstellung bleibt. Im Anschluss daran erfolgt der Ausweichvorgang in der Bildmitte, wo beide beteiligt sind und geht dann über in den Steuerbereich für den rechten Motor. Der grau hinterlegte Teil der Abbildung 61 dient zur Darstellung der Parameter für das Starten (in schwarz eingezeichnet) beziehungsweise das Beenden (in grau eingezeichnet) des Ausweichvorganges. Dieser Abschnitt ist in beiden Bildern gleich. Die hellgraue Linie stellt die Grenze zum Zeichenbereich dar. An den Bildrändern werden keine Ausweichbewegungen durchgeführt, deshalb wurden hier die Parameter auf 0 gesetzt. Wie man der Abbildung entnehmen kann wird der Ausweichvorgang später ausgelöst, wenn die Objekte weiter von der Bildmitte entfernt sind. Liegen die Objekte in einem Bereich von 2 Objektbreiten um die Bildmitte so wird der Ausweichvorgang noch vor der hellgrauen Linie, der Grenze zum Zeichenbereich, ausgelöst.

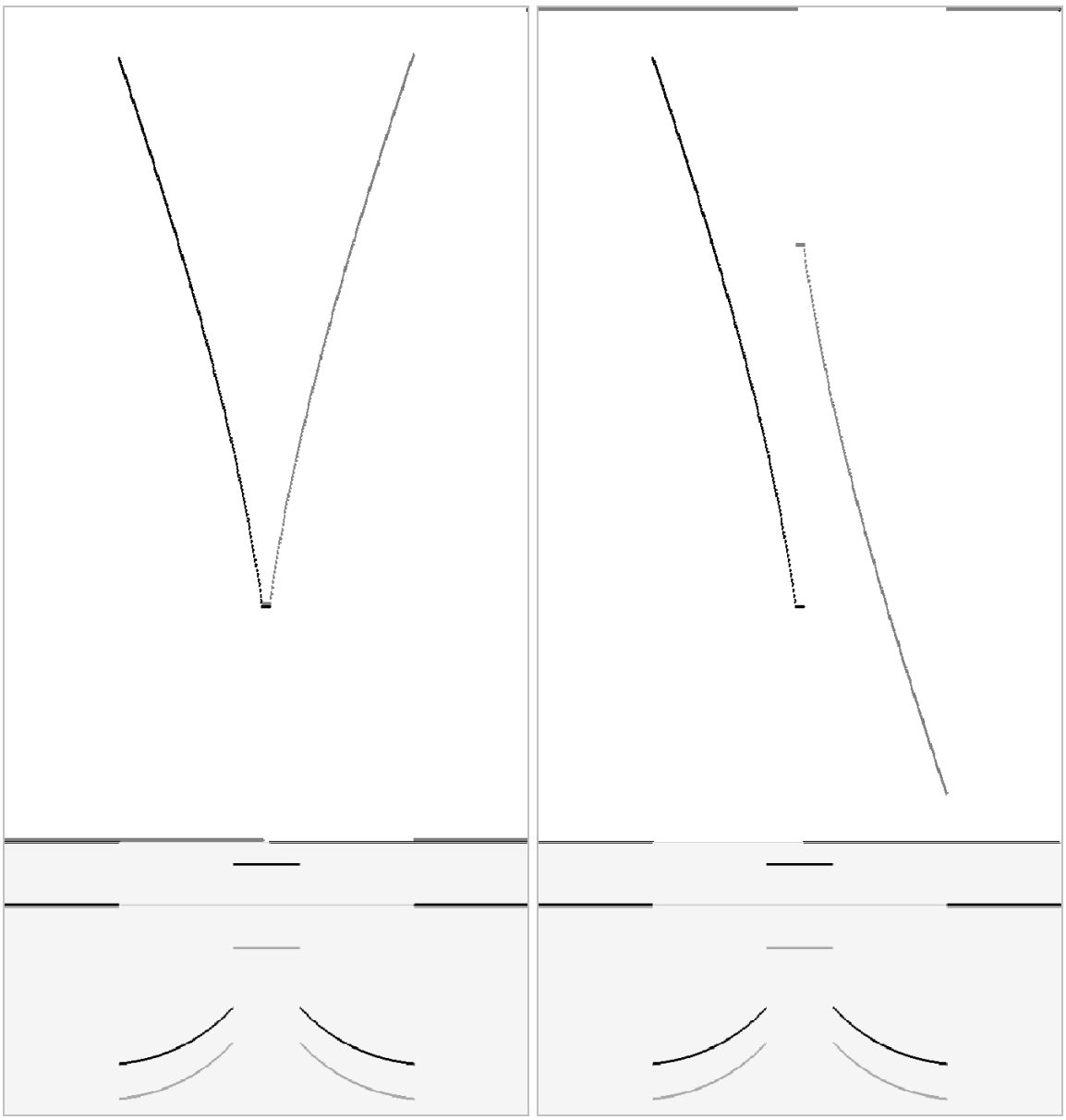


Abbildung 61: Positionsliste für den Ausweichvorgang bei Objektbreite von 20 Bildpunkten

5.6. Multithreading

Für die Ermittlung der Ausweichposition aus den Bilddaten sind einige Rechenschritte notwendig, welche in Abbildung 62 vereinfacht dargestellt sind. Am Anfang wird von der Kamera ein Farbbild aufgenommen, aus welchem im nächsten Schritt zuerst eine Grauwertmatrix berechnet und anschließend mit dem Verfahren von Otsu eine Binärmatrix erstellt wird. Diese wird dann im nächsten Unterprogramm segmentiert und eine Matrix erzeugt, welche die markierten Segmente beinhaltet. Der darauf folgende Programmteil errechnet aus der Segmentmatrix, die Objekteigenschaften und speichert diese in einem Feld. Dieses ist im Vergleich zu der Binär- und Segmentmatrix deutlich kleiner, da es nur noch einzelne Objekte enthält, die bereits mittels der Farbe und Fläche gefiltert wurden. Anschließend erfolgt die Registrierung, wobei die Felder mit den Objekteigenschaften zweier

aufeinanderfolgender Bilder miteinander verglichen werden und ein Verschiebungsvektor berechnet wird. Die zuletzt berechneten Objekteigenschaften werden gespeichert, da sie für die Registrierung des nächsten Bildes als Referenzdaten benötigt werden. Mit dem Verschiebungsvektor werden die Objekte außerhalb des Bildbereiches der Kamera verfolgt, bis sie von Auslegern der Motoren umfahren wurden. Beim Starten des Ausweichvorganges wird die Objektkoordinate an das Hauptprogramm zurückgeliefert. Diesem Wert wird eine Position beziehungsweise die Positionen für den Ausweichvorgang der Motoren zugeordnet. Zuletzt erfolgt der Aufruf einer Funktion welche die Positionswerte an die Frequenzumrichter schickt.

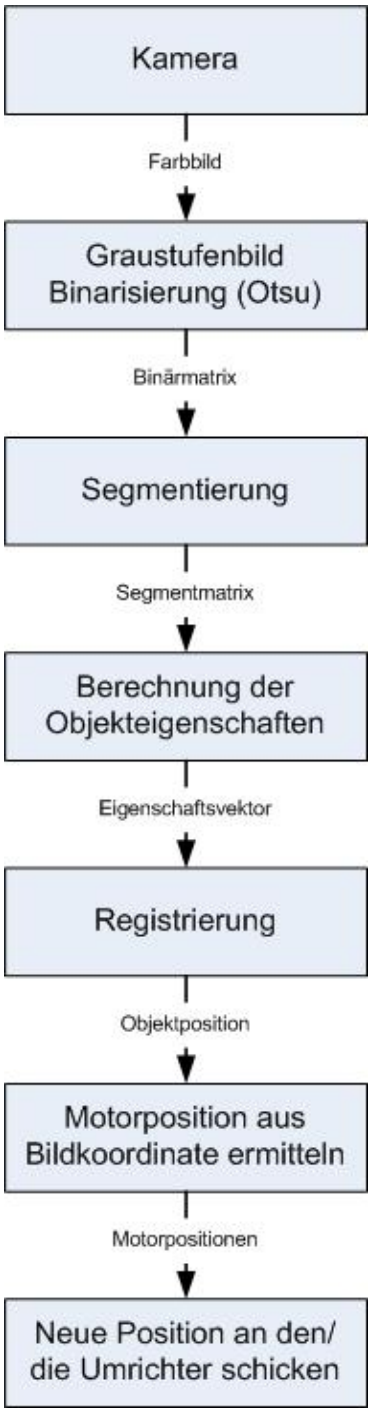


Abbildung 62: Programmteile

Diese Programmschritte müssen für jedes Bild durchlaufen werden, wobei vor allem die ersten drei Programmteile besonders rechenintensiv sind. Für die Binarisierung muss das Bild mehrmals Pixel für Pixel durchlaufen werden. Ebenso benötigt die Segmentierung einen Durchlauf durch die gesamte Binärmatrix, abgesehen von der Kantenverfolgung. Um die Objekteigenschaften zu gewinnen muss die Segmentmatrix ebenfalls einmal vollständig durchlaufen werden. Bei einem Bild mit 640x480 Bildpunkten sind das 307200 Elemente pro Bild beziehungsweise Matrix.
Die Ausführung der einzelnen Programmteile nacheinander nimmt sehr viel Rechenzeit in Anspruch und wurde deshalb sinnvoll auf mehrere Threads aufgeteilt.
Multithreading hat den Vorteil, dass mehrere Unterprogramme gleichzeitig abgearbeitet werden können beziehungsweise falls der Rechner mehrere Prozessoren besitzt, können rechenaufwendige Aufgaben auf die einzelnen CPU-Kerne aufgeteilt werden. Dabei ist darauf zu achten, dass eine sinnvolle Zuordnung der Einzelaufgabe erfolgt, da sonst der Gewinn durch die parallele Abarbeitung beim Warten auf Ergebnisse verloren gehen könnte.
Ein weiterer wichtiger Punkt ist, dass man bei der Verwendung von globalen Variablen vorsichtig sein muss, damit nicht mehrere Threads gleichzeitig dieselben Variablen benötigen beziehungsweise verändern. Die Verwendung von globalen Variablen ist notwendig, um die Ergebnisse einzelner Threads an einen anderen Thread weitergeben zu können. Einerseits ist es wichtig, die einzelnen Variablen zweckmäßig auf die Threads aufzuteilen und andererseits nach der Abarbeitung einer Aufgabe zu warten, bis die ermittelten Daten von dem nachfolgenden Thread übernommen werden können. Dies ist besonders für diese Anwendung wichtig, da sonst die Bildreihenfolge durcheinander gebracht werden würde und einzelne Bilder irgendwann zwischen den einzelnen Threads verloren gehen könnten. Wenn die Binarisierung schneller ist als die Segmentierung, aber nach dem Binarisieren eines Bildes nicht gewartet wird, bis die Segmentierung das neue Binärbild übernommen hat, dann kann es sein, dass manchmal jedes oder aber nur jedes zweite oder dritte Bild segmentiert wird.
Um den Vorteil mehrerer Threads nutzen zu können, ist eine sinnvolle Aufteilung der einzelnen Programmteile notwendig. Abbildung 63 zeigt die Realisierung des Multithreading-Ansatzes, durch das Aufteilen der Programmteile in drei Abarbeitungsstränge, die in etwa den gleichen Zeitaufwand benötigen. Ebenso wichtig bei der Aufteilung war eine zweckmäßige Nutzung von globalen und lokalen Variablen, wobei Erstere dafür benötigt werden um Werte und Felder zwischen den einzelnen Threads weiterzureichen.
Von dem Hauptprogramm aus werden die notwendigen globalen Variablen erzeugt und alle drei Threads gestartet. Um die Resultate der einzelnen Bildverarbeitungsschritte zwischen den einzelnen Threads weitergeben zu können werden die Bildnummer, ein Feld für die Binärmatrix, ein Feld mit den Motorpositionen und ein Feld für die Objekteigenschaften als globale Variablen benötigt. Die erste Variable wird für das Speichern der Ergebnisbilder der einzelnen Bildverarbeitungsschritte verwendet. Außerdem wird damit überprüft ob die Bildreihenfolge eingehalten wird und keine Bilder zwischen den Threads verlorengehen. Somit startet Thread 1 mit der Bearbeitung des Bildes mit der Nummer 5 erst wenn Thread 2 das Binärbild mit der Nummer 4 übernommen hat. Das Feld mit den Motorpositionen wird beim Starten des Programms generiert, sodass es einerseits von mehreren Programmteilen genutzt werden kann, beziehungsweise da es nur von Thread 3 benötigt wird, nicht bei jedem neuen Bild berechnet werden muss. Dies würde den Geschwindigkeitsvorteil Werte aus einer Tabelle auszulesen gegenüber deren Berechnung zunichte macht. Des Weiteren werden noch einige Variablen vom Typ Boolean benötigt, welche das Warten der einzelnen Threads steuern, beziehungsweise die Übernahme der globalen Variablen bestätigen.
Thread 1 beinhaltet die Bildaufnahme, wobei die Kamera ein Farbbild liefert. Anschließend wird die Funktion „Otsu()“ aufgerufen, welche aus dem Farbbild eine Binärmatrix erzeugt. Ebenso wird überprüft, ob das Eingangsbild, das berechnete Graustufenbild oder das Binärbild gespeichert werden soll. Das Resultat dieses Threads ist eine Binärmatrix, welche

global definiert wurde und die Eingangsgröße für die Segmentierung ist. Aus diesem Grund wird mit der Abarbeitung des nächsten Bildes gewartet bis der Thread 2 fertig ist und die eben berechnete Binärmatrix übernehmen kann. Wurde dies erledigt so setzt Thread 2 die Variable „NewOtsu“ auf den Wert „false“ und signalisiert so Thread 1, dass er mit der Bildaufnahme und Binarisierung des nächsten Bildes beginnen kann.
Thread 2 erledigt die Segmentierung der Binärmatrix sowie die Berechnung der Objekteigenschaften. Zuerst erfolgt die Überprüfung, ob ein Bild mit den markierten Segmenten erstellt werden soll und falls ja, wird aus der Bildnummer der Dateiname generiert. Im Anschluss daran wird die Funktion „ConnectedComponentLabeling()“ aufgerufen, welche die Binärmatrix segmentiert und danach werden mit der Funktion „GetFilteredLabelFeatures()“ die Eigenschaften der Segmente ermittelt. Zunächst erfolgt eine Speicherung der Objekteigenschaften in einem lokalen Feld und nach deren Berechnung wird gewartet, bis die Registrierung das Feld mit den globalen Objekteigenschaften abgearbeitet hat. Dies wird durch die Variable „RegistrierungFertig“ signalisiert. Solange diese den Wert „false“ hat, wird Thread 2 angehalten. Ist sie jedoch mit dem Wert „true“ beschrieben, dann werden die Objekteigenschaften von dem lokalen Feld auf das globale Feld kopiert. Anschließend wird die Variable „LabelingFertig“ auf „true“ gesetzt und signalisiert so den beiden anderen Threads, dass einerseits die Objekteigenschaften zur Verfügung stehen und andererseits eine neue Binärmatrix übernommen werden kann.

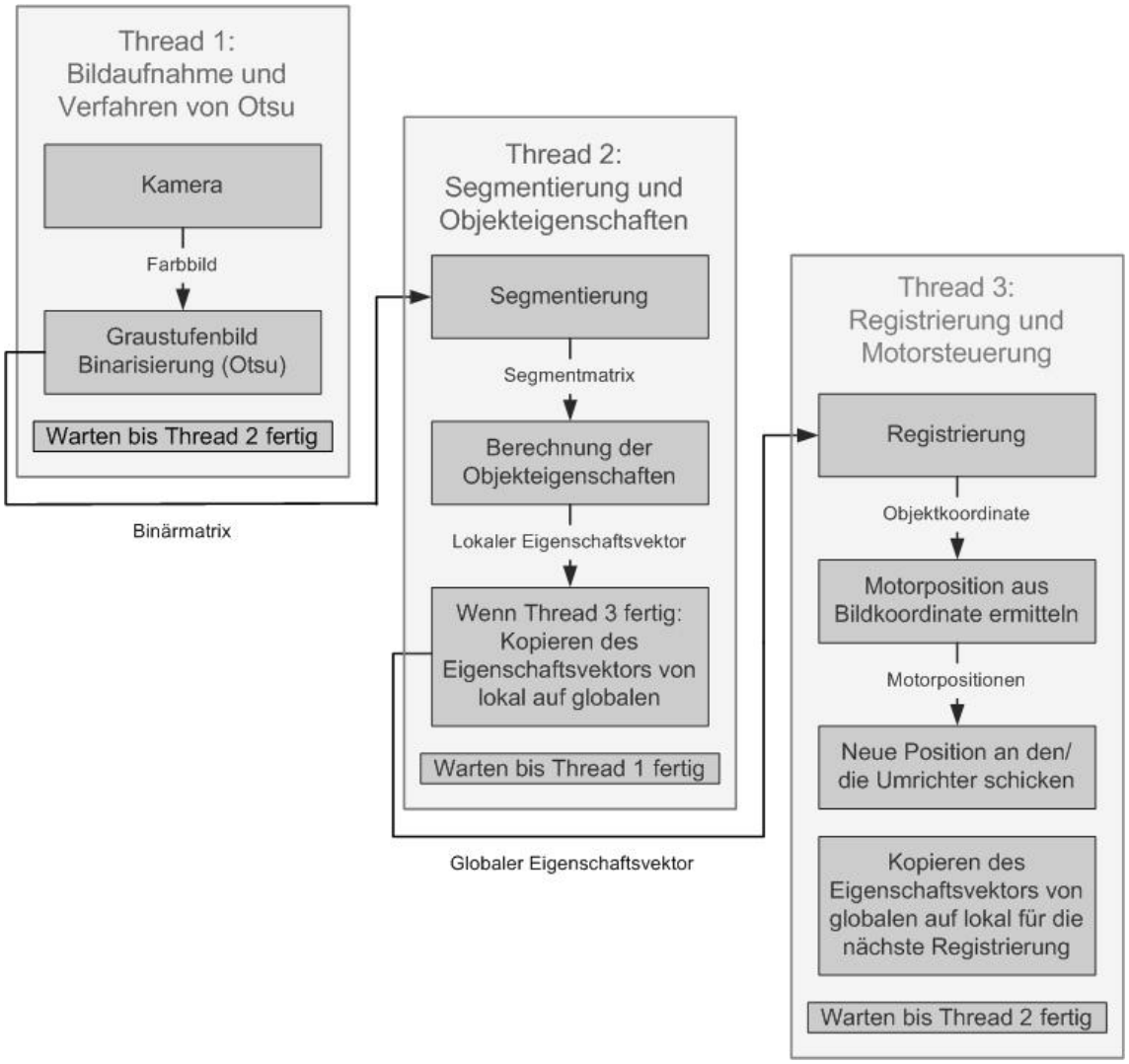


Abbildung 63: Multithreading

Der letzte Thread berechnet die Verschiebungsvektoren und speichert Objekte außerhalb des sichtbaren Bereiches in der Objektliste. Diese ist ein lokales Feld, ebenso wie die Objekteigenschaften des letzten Bildes, welche für die Registrierung benötigt werden. Dieser Thread ist für die Ausweichvorgänge und somit auch für die Kommunikation mit den Frequenzumrichtern zuständig. Im Verhältnis zu den beiden anderen Threads ist der Rechenaufwand geringer, jedoch erfordert die Kommunikation mit den Frequenzumrichtern über Ethernet einen gewissen Zeitaufwand.
Wie auch in den beiden anderen Threads wird zuerst überprüft, ob ein Speichern der Ergebnisse erforderlich ist. Falls ja wird aus der Bildnummer der Name generiert und anschließend die Funktion „Verschiebungsvektor()“ aufgerufen. Diese liefert für den Fall, dass einem Objekt ausgewichen werden soll, die x-Koordinate eines Objektes zurück, anderenfalls einen Wert größer der Bildbreite, was signalisiert, dass die Ausleger in Mittelstellung bleiben beziehungsweise bis in die Bildmitte schwingen können.
Anschließend wird diesem Rückgabewert eine Motorposition zugeordnet. Bei einem Ausweichvorgang werden die Werte aus der Motorpositionsliste ausgelesen, sonst erfolgt für jeden Frequenzumrichter eine Zuweisung der Mittelposition. Im Anschluss daran werden die Einstellungen des Hauptprogramms überprüft. Dort ist es möglich zwischen dem Betrieb mit einem oder beiden Motoren zu wählen, ebenso kann bei dem Betriebsmodus zwischen Schwingen und Nichtschwingen gewählt werden. Entsprechend dieser Auswahl erfolgt dann eine Positionsänderung von einem Motor oder von beiden Motoren durch den Aufruf der Funktion „IndexPositionSchreiben()“ für den Betriebsmodus Schwingen, anderenfalls „NeuePositionSchreiben()“. Für die Kommunikation mit den Frequenzumrichtern muss auch eine Fehlerbehandlung erfolgen, falls die Kommunikation nicht aufgebaut werden kann oder ähnliches. Deshalb werden die Funktionen in einem try-catch Block aufgerufen, um auftretende Exceptions abzufangen und eine Fehlermeldung zu erzeugen. Nach dem Schreiben der neuen Position auf den oder die Frequenzumrichter werden die Objekteigenschaften in ein lokales Feld kopiert, da sie zur Berechnung des nächsten Verschiebungsvektors benötigt werden, und die Variable „RegistrierungFertig“ auf „true“ gesetzt, um Thread 2 zu signalisieren, dass neue Werte auf das globale Feld für die Objekteigenschaften geschrieben werden können.
Das Hauptprogramm aus dem die einzelnen Threads gestartet werden können, besteht aus einer Windowsform, welche mehrere Steuerelemente enthält. Dadurch ist es möglich Werte einzugeben oder festzulegen ob ein oder zwei Motoren genutzt werden sollen. Diese Werte können von den einzelnen Threads abgefragt werden, jedoch ist es nicht möglich, aus einem Thread auf der Windowsform des Hauptprogramms den Wert einer TextBox zu verändern. Für die Entwicklung war es notwendig zu wissen, welches Bild gerade bearbeitet wird und ob die einzelnen Threads auch an den entsprechenden Stellen aufeinander warten. Dazu wurden Delegatenobjekte benötigt, welche einen Zugriff auf die Windowsform möglich machen [1].
In Abbildung 64 ist der Code für das Schreiben der aktuellen Bildnummer in die TextBox „textBoxZaehler“ abgebildet. Zuerst ist es notwendig einen Delegatentypen zu definieren. In diesem Fall handelt es sich um eine Methode mit dem Rückgabetypen void, welche einen einzigen Parameter besitzt. Dann wird eine Methode definiert, welche die TextBox leert und anschließend ihren Übergabewert dort hineinschreibt. Die Methode „BackgroundWorkerUpdateBildnummer()“ hat als Übergabeparameter die Nummer des aktuellen Bildes. Zuerst wird in dieser Methode ein Delegatenobjekt vom Typ DelegateUpdate erzeugt, welches auf die Methode „UpdateTextboxZahler()“ verweist. Danach wird durch „textBox.Zahler.Invoke()“, welcher das Delegatenobjekt und die zu schreibende Variable übergeben wird, ermöglicht, den Wert in die TextBox zu verändern. Durch den Aufruf der Methode „BackgroundWorkerUpdateBildnummer()“, mit der entsprechenden Nummer als Übergabeparameter, kann von einem Thread heraus die Bildnummer auf der Windowsform des Hauptprogramms geändert werden.

In der Windowsform wurde für jeden Thread eine TextBox eingefügt, welche Informationen darüber liefert, ob ein Thread läuft oder sich in Warteposition befindet. Jeder Thread besitzt einen Timer, der bei jedem neuen Bild gestartet wird und dessen Wert nach Beendigung der Bearbeitung an das Hauptprogramm geschickt wird. Damit kann überprüft werden, ob die Wartebedingungen der einzelnen Threads funktionieren. Ebenso können dadurch unterschiedliche Softwarerealisierungen miteinander vergleichen werden, wie etwa das Auslesen der Farbwerte eines Bildpunktes mit der C# Funktion „GetPixel()“ oder mit einem Zeiger. Funktioniert das Programm, so ändern die TextBoxen sehr schnell ihren Wert, wobei sich die Werte der ersten beiden Threads nur geringfügig ändern. Kann zu einem Frequenzumrichter keine Verbindung aufgebaut werden, so wird eine Fehlermeldung ausgelöst und der dritte Thread kann nicht fortgesetzt werden. Dies hat zur Folge, dass auch die beiden anderen Threads bis zum Wartepunkt laufen und dort verharren.

```
private delegate void DelegateUpdate(int value);

private void UpdateTextboxZaehler(int value)
{
        textBoxZaehler.Clear();
        textBoxZaehler.Paste(value.ToString());
}

private void BackgroundWorkerUpdateBildnummer(int Bildnummer)
{
        DelegateUpdate UpdateBildnummer =  new DelegateUpdate(UpdateTextboxZaehler);
        textBoxZaehler.Invoke(UpdateBildnummer, Bildnummer);
}
```

Abbildung 64: Update einer TextBox aus einem Thread

6. Die Testumgebung/Bedienoberfläche des Demonstrators

Da für die Lösung der Aufgabenstellung das Zusammenspiel von Bildverarbeitung, Steuerung der Servomotoren sowie der Ethernetkommunikation wichtig ist, wurde eine Windowsform erstellt, die es ermöglicht einzelne Komponenten aber auch das Gesamtsystem zu testen.

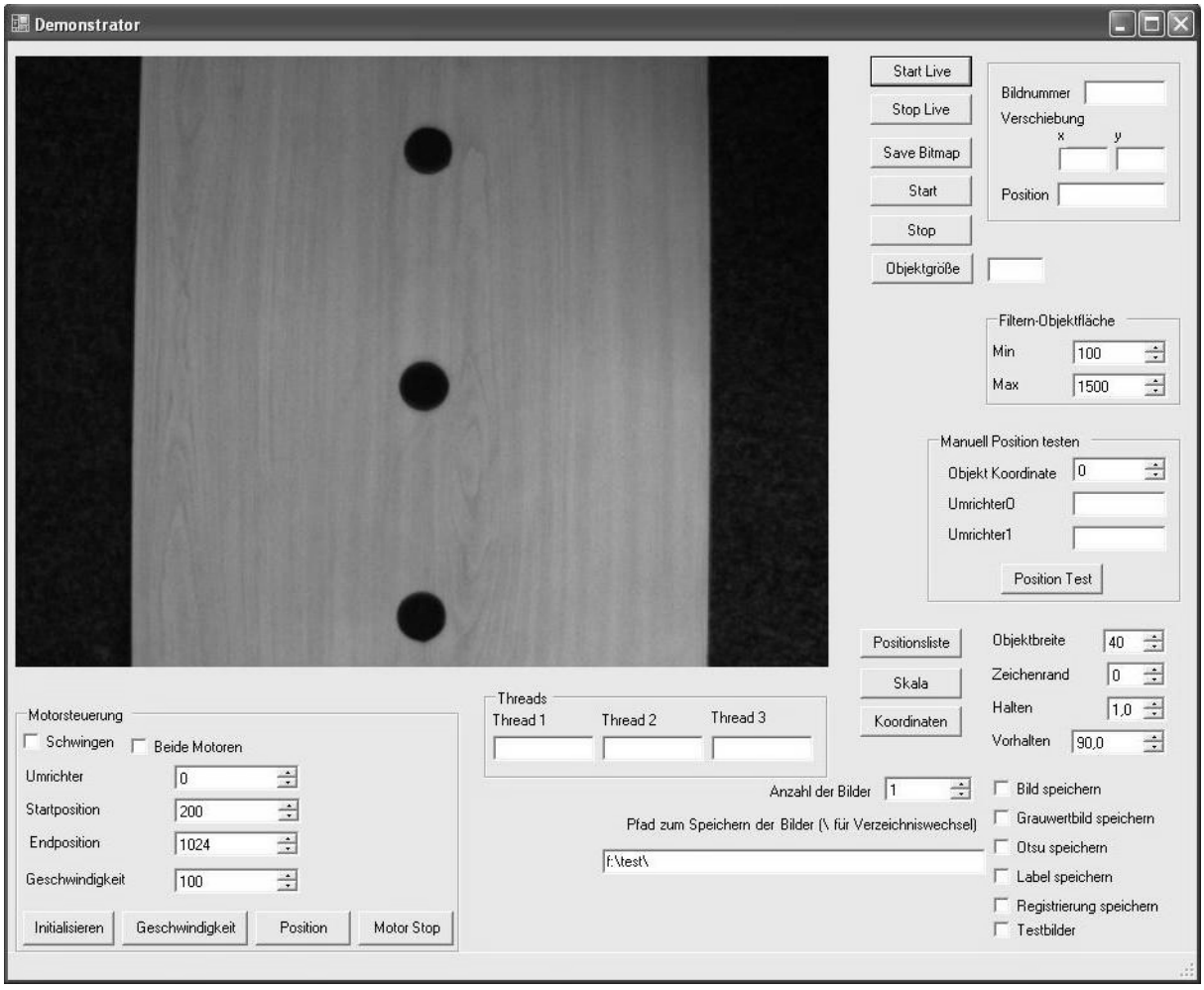


Abbildung 65: Die Bedienoberfläche/Windows Form

Abbildung 65 zeigt die Oberfläche der implementierten Testumgebung. Die Implementierung erfolgt mit C# in Microsoft Visual Studio 2005. Die Testumgebung bietet eine Vielzahl von Möglichkeiten zum Testen unterschiedlicher Parameter und Funktionalitäten, und bietet auch die Möglichkeit die Resultate der Bildverarbeitung mit sämtlichen Zwischenschritten zu speichern. Ebenso ist es möglich die Bilder der Kamera zu speichern, wodurch einerseits die Resultate des Ausweichvorganges durch ein nochmaliges Befahren der Strecke aufgezeichnet werden können, andererseits kann damit eine Bildserie zum Testen der Bildverarbeitungsalgorithmen aufgenommen werden.
Dominiert wird die Form durch das Kamerabild, welches Bilder der Strecke und der Hindernisse liefert. Rechts neben dem Sichtbereich der Kamera befinden sich Knöpfe, die in Abbildung 66 zusammengefasst wurden. Mit dem Knopf Start Live wird die Kamera im Live Modus aktiviert. Dieser aktiviert die Kamera und das Kamerabild wird auf dem Bildschirm dargestellt, jedoch werden keinerlei Bilddaten gespeichert.
Der Knopf Stop Live beendet den Live Modus und auf dem Bildschirm bleibt das letzte Bild erhalten. Das Tool zur Ausgabe des Kamerabildes, sowie Funktionen zum Starten und

Stoppen des Live-Modus oder zum Speichern des aktuellen Bildes der Kamera werden vom Hersteller zur Verfügung gestellt und können in die Windows Form eingebunden werden.

Abbildung 66: Bedienknöpfe des Demonstrators

Mittels Save Bitmap ist es möglich die Bilder der Kamera zu speichern. Für den Fall, dass sich die Kamera nicht im Live Modus befindet wird dieser Modus aktiviert und 500ms gewartet, bis das Kamerabild vollständig aufgebaut wurde. Abbildung 67 zeigt die Einstellungen, die beim Speichern des Kamerabildes gewählt werden können. Dadurch ist es möglich die Anzahl der Bilder, den Speicherort und die Art der Bilder einzustellen. Die Aufnahme der Bilder erfolgt alle 500ms und die Bilder werden unter dem angegebenen Pfad mit der jeweiligen Bildnummer gespeichert. Wenn Bilder direkt mit Save Bitmap gespeichert werden, so kann neben dem Kamerabild, ein Grauwertbild, ein Binärbild und ein Bild mit markierten Segmenten gespeichert werden.

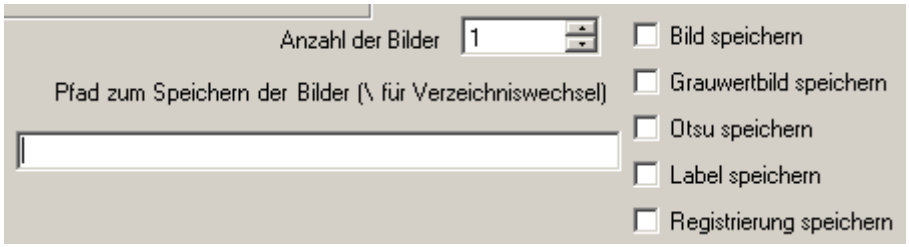


Abbildung 67: Einstellungen für Speicherung

Beim Speichern der Bilder und auch während des eigentlichen Programmablaufs wird zuerst abgefragt ob Bilder gespeichert werden sollen. Dazu wird überprüft ob in der TextBox aus Abbildung 67 ein gültiger Pfad eingetragen worden ist. Anschließend wird abgefragt, welche Bilder gespeichert werden sollen und die Dateinamen aus dem Pfad, der Bildnummer und einer Extension zusammengefügt und der jeweiligen Funktion übergeben. Die Bilder werden mit der gleichen Bildnummer gespeichert, nur wird zur Unterscheidung vor der Extension _g für ein Grauwertbild, _o für ein Binärbild und _l für ein Segmentbild eingefügt.
Abbildung 68 zeigt das aus dem Farbbild berechnete Grauwert-, Binär- und Segmentbild.

Abbildung 68: Graustufenbild (_g), Binärbild (_o), Segmentbild (_l)

Mit dem Knopf Start aus Abbildung 66 wird das eigentliche Programm zur Objekterkennung und Umfahrung gestartet. Zuerst wird überprüft, ob die Kamera aktiv ist, falls nicht wird diese aktiviert. Anschließend wird die Liste der Motorpositionen für die eingestellte Objektbreite berechnet. Danach erfolgt die Initialisierung der Frequenzumrichter durch den Aufruf der Funktionen für die Initialisierung. Im Anschluss daran werden die Threads gestartet. Es ist auch möglich, während der Laufzeit des Programms Änderungen vorzunehmen, beispielsweise welche Bilder gespeichert werden sollen oder die Änderung der Positioniergeschwindigkeit. Änderungen der Objektgröße werden nicht berücksichtigt, da diese für die Motorpositionen benötigt werden und die Liste vor dem Start berechnet und danach nicht mehr aktualisiert wird.
Es ist aber auch möglich, durch das Markieren der CheckBox Testbilder den Programmablauf an einer Bildserie zu testen. Dazu werden Bilder aus dem Speicher geladen und diese durchlaufen sämtliche Verarbeitungsschritte. Dies war vor allem für die Entwicklung wichtig, kann aber auch dazu genutzt werden, um zu testen, ob der Demonstrator unter den gegebenen Licht- und Umgebungsverhältnissen überhaupt arbeiten kann. Wird mit Testbildern gearbeitet, so werden die Frequenzumrichter nicht benötigt, weshalb diese nicht initialisiert werden und auch keine Kommunikation erfolgt. Dadurch ist es möglich, die Bildverarbeitung auch auf Rechnern zu testen, die nicht auf dem Versuchsaufbau moniert sind.
Der Knopf Stop beendet den Ablauf des Programms indem er durch den Aufruf der Funktion „StopIndex()“ die Motoren stoppt und die Threads beendet.
Durch Drücken des Knopfes Objektgröße wird das aktuelle Kamerabild binarisiert und segmentiert. Anschließend werden die Eigenschaften der Segmente ermittelt und gefiltert. Die Eigenschaften des Filters können über die NumericUpDown-Steuerelmente eingestellt werden, wie weiter unten beschrieben. Dies ermöglicht, dass die dunklen Ränder oder Objekte mit unerwünschter Größe bei der Berechnung nicht berücksichtigt werden. Hinterher wird die größte Objektfläche ermittelt und unter der Annahme, dass es sich um ein rundes Element handelt, der Durchmesser berechnet. Zu diesem Wert werden noch 20 Pixel addiert damit bei der Ausweichbewegung genügend Platz zum jeweiligen Objekt bleibt. Die Berechnung des Mittelwertes aller gefundenen Objekte ist nicht ratsam, da es passieren kann, dass einzelne Objekte nur zum Teil im Sichtbereich der Kamera liegen, was das Ergebnis verfälschen würde. Zuletzt wird in der TextBox neben dem Knopf die Objektgröße eingetragen und der Wert für die Objektbreite in dem gleichnamigen NumericUpDown-Steuerelement gesetzt. Dadurch werden die Motorpositionen für diese Objektgröße berechnet.
Der Knopf Positionsliste berechnet die Liste der Motorpositionen für den eingestellten Objektdurchmesser und gibt eine MessageBox mit den vier Parametern für jede x-Koordinate aus, die angefahren werden kann. Ebenso wird ein Bild, wie in Abbildung 61, erstellt, das die Motorpositionen und den Start- und Endwert des Ausweichvorganges darstellt.
Mit dem Knopf Skala wird aus der Positionsliste eine Skala, siehe Abbildung 69, generiert und unterhalb des Kamerabildes eingeblendet.

Abbildung 69: Diese Skala wird unter dem Kamerabild eingeblendet

Die dunkelgrauen Ränder zeigen den Bereich an, in dem die Objekte nicht umfahren werden. In der Mitte ist ein weißer Balken, welcher der eingestellten Objektgröße entspricht und einerseits für den Vergleich genutzt werden kann und andererseits zur Orientierung dient. Der Grauverlauf wird aus den Motorpositionen berechnet und zeigt den Wirkungsbereich der einzelnen Frequenzumrichter an.
Um die Koordinaten einzelner Objekte bestimmen zu können dient der Knopf Koordinaten. Dafür wird das aktuelle Kamerabild binarisiert und segmentiert. Anschließend werden die Objekteigenschaften ermittelt und gefiltert, wobei auch hier die Einstellungen des Filters berücksichtigt werden. Von allen Objekten, die nicht herausgefiltert wurden, werden, wie in Abbildung 70 dargestellt, die Koordinaten des Schwerpunktes und die Fläche in einer MessageBox ausgegeben.

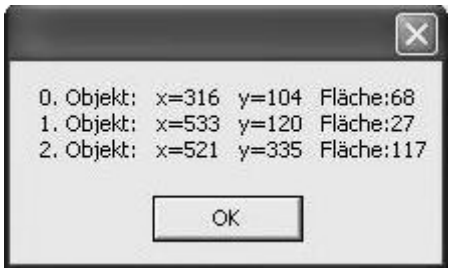


Abbildung 70: Objektkoordinaten und Fläche

Diese Funktion wurde genutzt um die Motorachsen, die Länge der Ausleger, den Abstand zwischen der Unterkante des Sichtbereiches der Kamera sowie die Ausrichtung der Kamera zu erfassen.
Um die Funktion der Registrierung zu überprüfen wurden die TextBoxen aus Abbildung 71 in die Windowsform eingebunden. Sie informieren über die aktuelle Bildnummer, die x- und y-Komponente des Verschiebungsvektors, sowie über die x-Position des Objektes, welchem gerade ausgewichen wird. Für den Fall, dass es kein Ausweichvorgang stattfindet, wird der Wert der Bildmitte angezeigt.

Abbildung 71: Ausgabe der Registrierung

Um unterschiedliche Objekte detektieren zu können, ist es möglich die für die Filterung notwendige Mindest- und Maximalfläche mittels NumericUpDown-Steuerelementen einzustellen, siehe Abbildung 72.

Abbildung 72: Für die Filterung können Minimal- und Maximalfläche eingestellt werden

Damit das Zusammenspiel von Servomotoren und Motorpositionsliste überprüft werden kann, ist es möglich für jede x-Koordinate die Servomotoren auszurichten. Durch das Einstellen der x-Koordinate im NumericUpDown-Steuerelement aus Abbildung 73, werden aus der Positionsliste die Motorpositionen ausgelesen und an die Frequenzumrichter geschickt, beziehungsweise in den TextBoxen ausgegeben. Für die Positionierung der Servomotoren wird nur ein Positionsindex verwendet, damit die Position der Ausleger besser überprüft und beim Weiterschalten der x-Koordinate auch die Änderung leichter nachvollzogen werden kann.

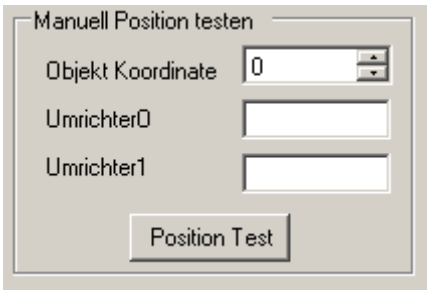


Abbildung 73: Manueller Positionstest

Weiters gibt es die Möglichkeit mit NumericUpDown-Steuerelementen die Objektbreite, den Zeichenrand, den Startpunkt sowie das Halten der Ausweichbewegung manuell zu verstellen, siehe Abbildung 74. Dadurch ist es möglich für verschiedene Positioniergeschwindigkeiten und Fahrgeschwindigkeiten Einstellungen zu testen um gute Resultate zu erzielen.
Der Parameter Zeichenrand verändert den Abstand von der Unterkante des Kamerabildes bis zum Zeichenbereich. Mit dem Parameter Halten kann eingestellt werden, wie lange die Ausweichposition gehalten werden soll. Dieser Parameter wird als Vielfaches der Objektbreite eingestellt. Die Einstellung Vorhalten legt fest, wie viele Bildpunkte vor beziehungsweise nach dem berechneten Startzeitpunkt der Ausweichvorgang eingeleitet werden soll.

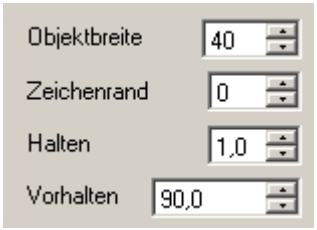


Abbildung 74: Einstellmöglichkeiten für den Ausweichvorgang

Für die Implementierung des Multithreading waren die Wartebedingungen der einzelnen Threads von großer Bedeutung. Deshalb wurde die in Abbildung 75 dargestellten TextBoxen in die Windowsform eingefügt. Bei der Übernahme eines neuen Bildes wird für jeden Thread ein Timer gestartet und nach der Abarbeitung gestoppt und der Zählerstand ausgegeben. Der

Zählerstand wird erst wieder gelöscht, wenn der Thread mit der Abarbeitung eines neuen Bildes beginnt, wofür das letzte Bild von dem nachfolgenden Thread übernommen werden muss. Dadurch konnte auch der Geschwindigkeitsunterschied beim Auslesen der Farbwerte mit den C#-Funktionen beziehungsweise den Zeigern sichtbar gemacht werden.

Abbildung 75: Timerstände der einzelnen Threads

Für das Testen der einzelnen Funktion der Motorsteuerung wurde die in Abbildung 76 dargestellten Steuerelemente implementiert. Diese ermöglichen zwischen den beiden Betriebsmodi und der Verwendung von einem oder beiden Servomotoren zu wählen. Mit diesen Parametern wird dann bestimmt, welche Motorfunktionen durch das Drücken der Knöpfe aufgerufen werden. Für den Positioniervorgang im Betriebsmodus Schwingen werden zwei Indizes für die Positionierung verwendet. Deshalb müssen bei der Initialisierung die Startposition und die Endposition übergeben werden. Für eine Positionsänderung ist es erforderlich, den Index anzugeben, dessen Position geändert werden soll. Ebenso müssen bei einer Geschwindigkeitsänderung die Geschwindigkeiten beider Indizes verändert werden. Wenn keine schwingenden Bewegungen erfolgen sollen, werden nur die Endposition und die zugehörige Geschwindigkeit benötigt. Die Änderung der Motorparameter kann für beide oder aber auch für einen Motor durchgeführt werden, auch wenn beide Motoren initialisiert wurden. Mit einem NumericUpDown-Steuerelement kann ein Frequenzumrichter ausgewählt werden, wodurch beim Aufruf der Funktion zur Parameteränderung die entsprechende IP-Adresse zuordnet wird. Für den rechten Servomotor werden die Positionen zusätzlich umgerechnet indem die eingestellten Positionen von 1024 subtrahiert werden.
Der Knopf Motor Stop stoppt den Motor an einer vorgegeben Position. Diese wurde für beide Motoren so festgelegt, dass die Ausleger in Fahrtrichtung stehen. Diese Funktion wurde implementiert, da ein Abschalten der Reglerfreigabe nur verhindert, dass die Motoren weiterhin geregelt werden, aber diese nicht in einem definierten Zustand zum Stillstand kommen.

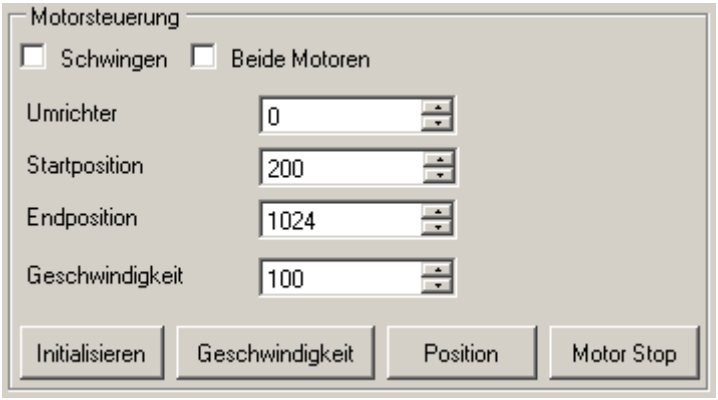


Abbildung 76: Motorsteuerung

7. Vermessung und Ausrichtung

Für die Berechnung der Objektpositionen ist es wichtig, dass die Kamera parallel zum Boden ausgerichtet ist, da sonst Verzerrungen auftreten. Daraus würden bei der Berechnung des Verschiebungsvektors Probleme resultieren, da die Verschiebung zwischen den Objekten des Modellbildes und des Referenzbildes eigentlich gleich sein müsste, aber aufgrund der Neigung der Kamera als unterschiedlich erkannt werden. Ebenso ist es wichtig, die Lage der Motorachsen, die Länge der Ausleger und den Abstand zwischen dem Bildbereich der Kamera und dem Zeichenbereich zu kennen. Diese Parameter ändern sich, da die Verwendung unterschiedlich langer Ausleger oder unterschiedliche Entfernungen der Kamera zur Motorachse (wegen der verschiebbaren Kameraaufnahme) möglich sind.

7.1. Abstandsermittlung

Da die Objekte, sobald sie den Bildbereich der Kamera verlassen, in der Objektliste gespeichert werden und deren Positionen dort solange weiterberechnet werden bis der Ausweichvorgang abgeschlossen ist, ist die Kenntnis des Abstandes zwischen der Unterkante des Sichtbereichs der Kamera und der Motorachse (die der Grenze zum Zeichenbereich entspricht) notwendig.
Um diesen Abstand zu ermitteln, wurde eine Markierung auf der Zeichenfläche angebracht, anschließend der Aufbau soweit verschoben bis diese an der Unterkante des Sichtbereichs der Kamera gerade noch zu sehen war. Im Anschluss daran wurden die Ausleger in die Mittelposition gebracht ohne eine Berührung der Stifte auf der Zeichenfläche zuzulassen. Auf Höhe der Spitzen der Stifte wurde ebenfalls eine Markierung angebracht und anschließend der Aufbau soweit zurück geschoben, bis beide Markierungen im Kamerabild zu erkennen waren. Durch Drücken des Knopfes Koordinaten wurden die Schwerpunkte der beiden Markierungen ermittelt. Abbildung 77 links zeigt das Kamerabild, das rechte Teilbild die ermittelten Schwerpunktskoordinaten. Daraus errechnet sich der Abstand zwischen den beiden Bereichen als Differenz der beiden y-Koordinaten und beträgt somit 460 Bildpunkte.

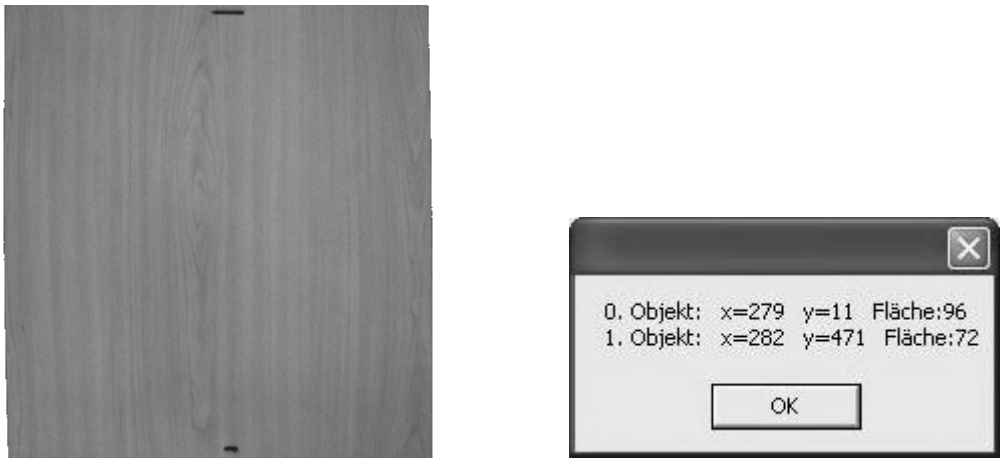


Abbildung 77: Messung des Abstandes zwischen dem Bildbereich und dem Zeichenbereich

7.2. Bestimmung der Motorachse und der Auslegerlänge

Die Länge der Ausleger und die Lage der Motorachse sind für die Berechnung der Motorpositionen des Ausweichvorganges notwendig und wurden für die aktuelle Gegebenheiten vermessen. Dafür wurde ein Viertelkreis, von Position 0 bis Position 1024 auf die Zeichenfläche gezeichnet und anschließend die Stifte von der Zeichenfläche abgehoben, damit diese frei über der Zeichenfläche bewegt werden können. Dies war notwendig, da zum Zeichnen der Linien ein Druck benötigt wird, wodurch einerseits die Ausleger geringfügig verbogen werden und andererseits der Regler etwas überschwingt, da er die Position nicht gleich beim ersten Anlauf genau anfahren kann. Da nun die Stifte knapp über der Zeichenfläche hängen, konnten deren exakte Positionen am Anfang und Ende des Positionsbereiches markiert werden. Der nicht benötigte Teil des Kreisbogens wurde entfernt. Die Position der Motorachse wurde mit einem Lot ermittelt und markiert. Anschließend wurden die Koordinaten der einzelnen Punkte ermittelt, siehe Abbildung 78.

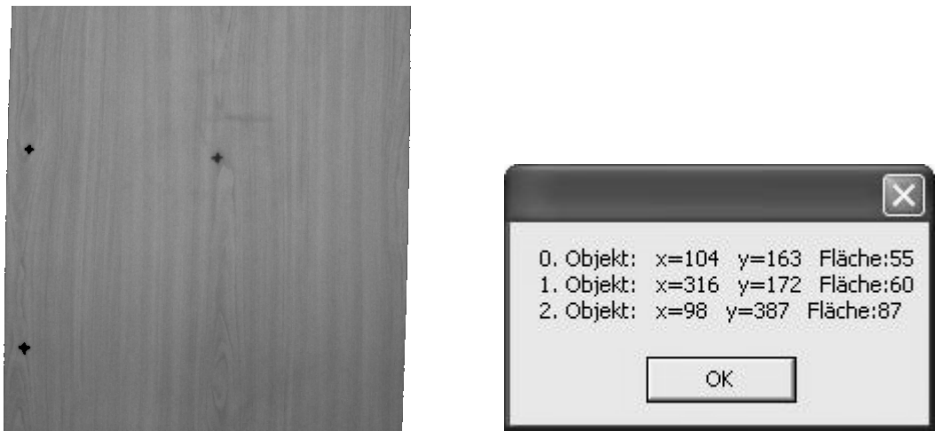


Abbildung 78:Koordinaten der Motorachse und des Auslegers bei Position 0 und 1024

Das rechte Teilbild von Abbildung 78 zeigt die Objektkoordinaten für die Motorachse und den Anfang- und Endpunkt der Ausweichbewegung des linken Servomotors. Objekt 0 ist die Markierung links oben im linken Teilbild und entspricht der Motorachse. Objekt 1 befindet sich in der Bildmitte und steht für die Position 1024 und Objekt 2 steht für die Position 0 und befindet sich links unten im Bild. Zur Bestimmung der Auslegerlänger r wird der Betrag des Vektors zwischen Objekt 0 und Objekt 1 beziehungsweise Objekt 2 benötigt. Für das Objekt 1 ergibt der Betrag

$$r_1 = \sqrt{(104-316)^2 + (163-172)^2} = 212 \qquad \textbf{(7.1)}$$

und für Objekt 2

$$r_2 = \sqrt{(104-98)^2 + (163-387)^2} = 224\,. \qquad \textbf{(7.2)}$$

Eigentlich sollten die Beträge der beiden Abstände gleich sein. Der Unterschied kann einerseits daraus resultieren, dass bei der Vermessung der Motorachse die Position des Lotes nicht exakt eingezeichnet wurde, andererseits besteht die Möglichkeit, dass durch die Fertigung oder die Montage der Motor schief eingebaut wurde und dadurch je nach Position mehr Druck auf den Stift ausübt und der Ausleger stärker verbogen wird.

Da die beiden Vektoren von der Motorachse M zu den Objekten O_1 und O_2 gleich lang sind und normal aufeinander stehen, wurden die Koordinaten der Motorachse und die Auslegerlänge wie folgt bestimmt:

$$M = O_1 - \vec{r} = O_2 - \vec{n}_r \qquad \textbf{(7.3)}$$

$$M = O_1 - \begin{pmatrix} x \\ y \end{pmatrix} = O_2 - \begin{pmatrix} -y \\ x \end{pmatrix}. \quad \textbf{(7.4)}$$

Setzt man für die Objekte ihre Koordinaten ein

$$\begin{pmatrix} 316 \\ 172 \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 98 \\ 389 \end{pmatrix} - \begin{pmatrix} -y \\ x \end{pmatrix} \quad \textbf{(7.5)}$$

so erhält man folgendes Gleichungssystem

$$\begin{aligned} 316 - x &= 98 + y \qquad \Rightarrow 218 - x = y \\ 172 - y &= 389 - x \qquad \Rightarrow 172 - (218 - x) = 389 - x \end{aligned} \quad \textbf{(7.6)}$$

Die Lösung des Gleichungssystems liefert x=217,5 und y=0,5. Daraus errechnen sich die Koordinaten der Motorachse als (98,5|171,5). Die Länge des Auslegers ist der Betrag des Vektors r und dieser beträgt 217,5. Der Mittelwert der beiden Abstände r_1 und r_2 ergibt 218 was dem gerundeten Betrag des Vektors entspricht. Die Differenz der gemessenen Koordinaten von den berechneten beträgt, auf ganze Bildpunkte gerundet, 5 in x- und 9 in y-Richtung. Der Vorgang wurde auch für die Motorachse und die Auslegerlänge des rechten Motors durchgeführt.

7.3. Ausrichtung der Kamera

Damit bei der Objektdetektion die Koordinaten der Objektschwerpunkte auch exakt bestimmt werden können ist es wichtig, dass das Kamerabild parallel zum Boden aufgenommen wird. Ist die Kamera hingegen etwas verschoben, so wird ein Teil des Kamerabildes gezerrt und der andere gestaucht. Dies ist vor allem bei der Berechnung des Verschiebungsvektors unerwünscht, da die Verschiebung der einzelnen Objekte vom Referenzbild zum Modellbild normalerweise annähernd gleich sind, bei einer falsch ausgerichteten Kamera jedoch stark voneinander abweichen können. Um festzustellen ob die Kamera parallel zum Boden ausgerichtet ist, wurde dort ein Blatt mit 9 schwarzen Punkten, welche an einem Gitter angeordnet wurden, hingelegt und anschließend die Koordinaten der Punkte bestimmt. Abbildung 79 zeigt im linken Teilbild das Kamerabild und im rechten Teilbild die berechneten Objektkoordinaten.

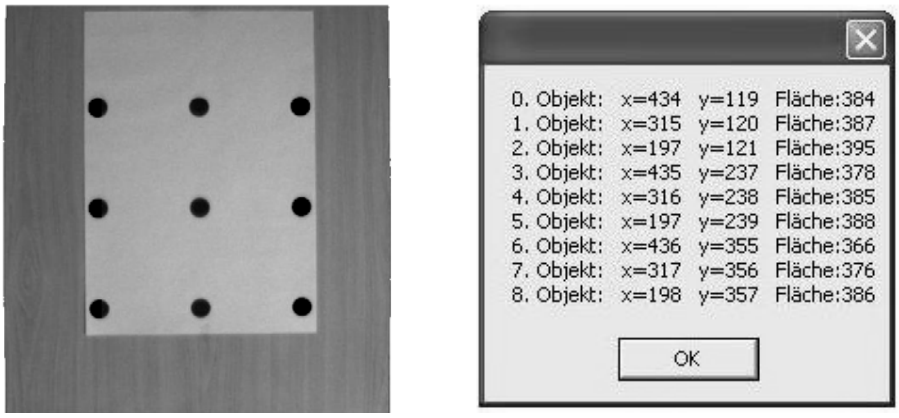


Abbildung 79: Ausrichtung der Kamera testen

Vergleicht man die y-Koordinaten der ersten drei Objekte miteinander, welche die oberste Reihe des Kamerabildes darstellt, so ist eine Differenz von einem Bildpunkt zum nächsten festzustellen. Da dies auch für die zweite und dritte Reihe zutrifft, ist es naheliegend, dass das Blatt nicht parallel zur Bildkante liegt. Die Differenz der x-Werte für die Objekte 0, 4 und 6 betragen auch nur einen Bildpunkt zum nächsten Nachbarn. Somit kann davon ausgegangen werden, dass die Kamera gut ausgerichtet ist und es zu keinen Verzerrungen im Bild kommt.

8. Ergebnisse

In diesem Kapitel werden die Resultate der Ausweichsteuerung präsentiert. Auf dünnen furnierten Pressspanplatten wurden runde Objekte mit ungefähr vier Zentimeter Durchmesser aufgemalt. An den Auslegern der Servomotoren wurden Whiteboard-Marker befestigt, um die Ausweichbewegungen sichtbar zu machen. Die Resultate der Testfahrten lieferten Informationen über den richtigen Startzeitpunkt, wie lange der Ausweichvorgang dauern muss und welcher Ausschlag notwendig ist, um ein Hindernis nicht zu treffen. Nachfolgend sind einige Bilder von Ausweichvorgängen mit unterschiedlichen Fahr- und Positioniergeschwindigkeiten sowie Betriebsmodi.
Am Ende der Testphase sind die Klemmungen für die Stifte an den Auslegern abgebrochen. Um weitere Tests zu ermöglichen, wurden Metallwinkel an die Ausleger geschraubt, um die Stifte daran fixieren zu können. Dies führte zu einer Erhöhung der Masse am Ende des Auslegers und verursachte dadurch ein Überschwingen. Die Frequenzumrichter konnten somit die Servomotoren nicht immer auf die richtige Position regeln. Beim Versuch den Motor abzubremsen, wird die Sollposition überschritten und es muss nochmals nachgeregelt werden. Aus dem zu hohen Masseträgheitsverhältnis resultiert teilweise starkes Überschwingen, wie an einigen der nachfolgenden Bildern zu sehen ist.
In den folgenden Bildern ist die Fahrtrichtung von unten nach oben. Ein neues Objekt erscheint im oberen Bereich des Bildes, durchwandert es und wird beim Überschreiten der Unterkante in die Objektliste aufgenommen.
Abbildung 80 zeigt die Ausweichbewegung für Objekte in der Bildmitte. Sowohl die Fahrgeschwindigkeit wie auch die Positioniergeschwindigkeit sind sehr gering gewählt, da die Ausweichbewegung sehr fließend verläuft und keinerlei Kanten aufweißt. Die Ausweichposition wird nur relativ kurz gehalten, da kurz nach dem Erreichen des Wertes der Ausleger wieder in Richtung Bildmitte schwingt.

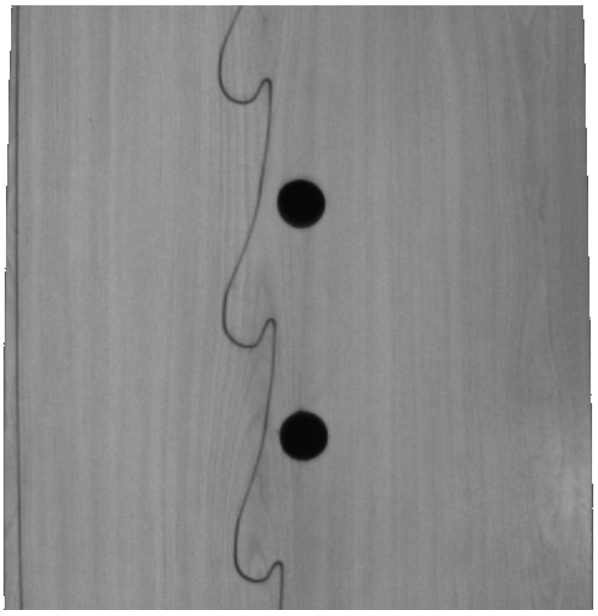


Abbildung 80: Linker Ausleger, Objekte in der Bildmitte

Im Gegensatz dazu wird in Abbildung 81 mit einer etwas höheren Geschwindigkeit positioniert. Für den Ausweichvorgang wird in diesem Fall nur der linke Ausleger benutzt. Da das Objekt über der Bildmitte liegt, wird der Ausweichvorgang erst eingeleitet, wenn der Stift bereits neben oder hinter dem Objekt ist. Nach dem Start des Ausholvorganges wird der Ausleger ausgelenkt, jedoch kann an der gewünschten Position nicht abgebremst werden und

es wird erst im Anschluss daran auf die eigentliche Position geregelt. Dies erkennt man sehr gut an der Ausweichbewegung für das mittlere Objekt, wo der Ausleger vom Maximalausschlag langsam auf seinen Sollwert geführt wird.

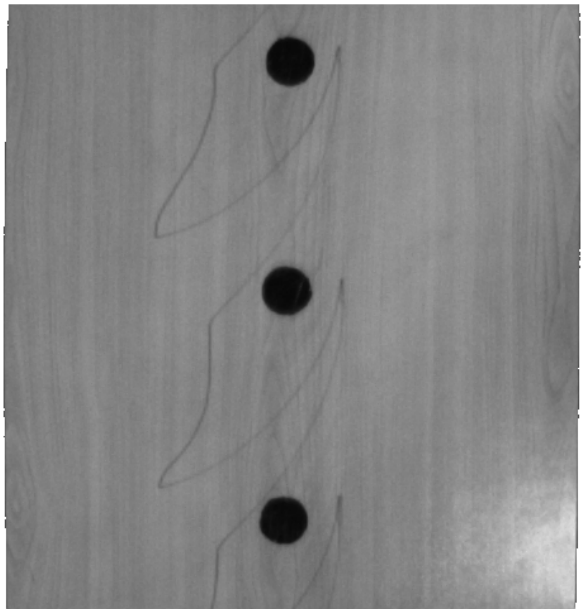


Abbildung 81: Linker Ausleger, leichtes Überschwingen

In Abbildung 82 erfolgt der Ausweichvorgang mit dem rechten Ausleger, wobei die Positioniergeschwindigkeit deutlich höher als im vorigen Bild ist. Der Ausweichvorgang startet wieder neben oder hinter dem Objektmittelpunkt, nur das hierbei das Überschwingen relativ deutlich zu sehen ist. Die Spitze entsteht durch die hohe Positioniergeschwindigkeit was im Gegensatz zum vorigen Bild ein nochmaliges Überschwingen zur Bildmitte hin verursacht bis die Sollposition erreicht wird. Die Objektgröße wurde hier relativ klein gewählt, wodurch der Stift sehr knapp um das Objekt herumfährt. Die Dauer des Ausweichvorganges wurde auf den dreifachen Objektdurchmesser festgelegt.

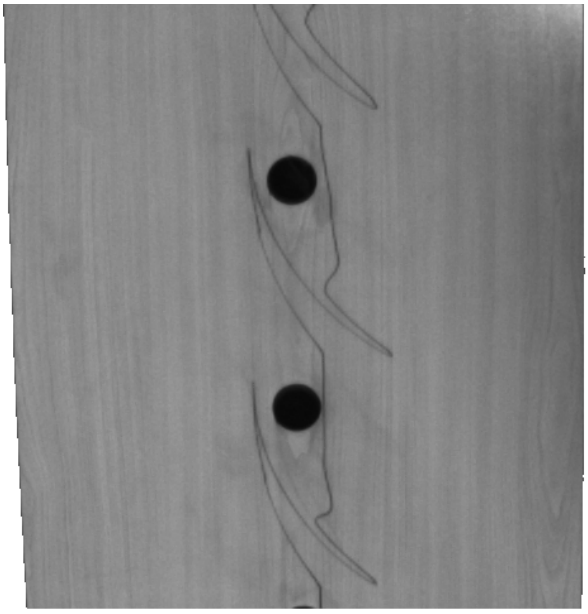


Abbildung 82: Rechter Ausleger

In Abbildung 83 erfolgt der Ausweichvorgang mit niedriger Positionier- und Fahrgeschwindigkeit. Die Dauer des Ausweichvorganges ist knapp gewählt, da die Ausleger schon in Mittelstellung gebracht werden, während sie sich noch neben dem Objekt befinden. Die Wahl der Dauer des Ausweichvorganges ist insofern schwierig, weil man die aktuelle y-Koordinate des Auslegers nicht kennt. Während der Ausleger auf Mittelposition gebracht wird, befindet sich die Grenze zum Zeichenbereich schon auf Höhe des nachfolgenden Objekts, und so muss gleich von der Mittelposition wieder die nächste Ausweichbewegung eingeleitet werden.

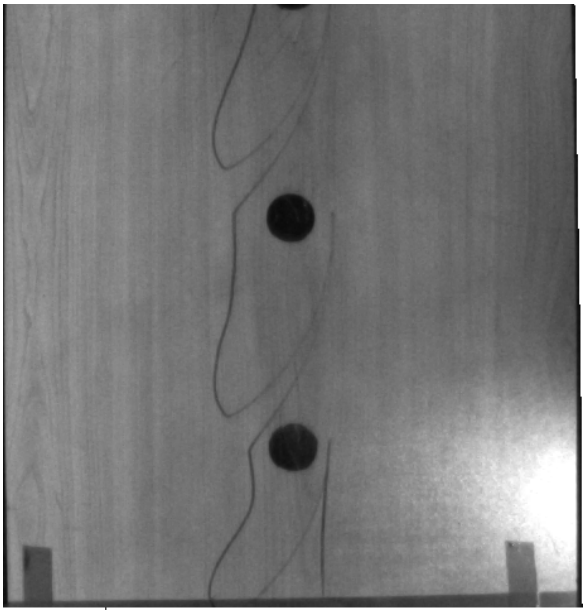


Abbildung 83: Linker Ausleger

In Abbildung 84 wurde mit dem linken Ausleger der Betriebsmodus Schwingen getestet. Die Positioniergeschwindigkeit ist relativ hoch, da die Ausholbewegung rasch erfolgt und die Schwingbewegungen dicht aneinander liegen. Der Ausweichvorgang wird zu früh eingeleitet und die Ausweichbewegungen sind deutlich zu groß.

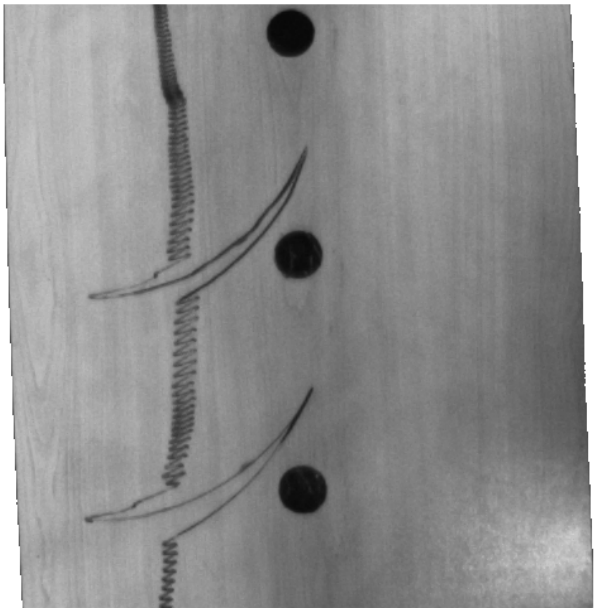


Abbildung 84: Linker Ausleger, Betriebsmodus Schwingen

Die Startposition für die Schwingbewegung wurde auf den kleinsten Wert der Motorpositionsliste gesetzt, damit im Falle eines Überschwingens noch genug Platz vorhanden ist und nicht über die Holzplatte hinausgemalt wird.
Die Ausweichbewegungen für zwei Servomotoren ist in Abbildung 85 dargestellt. Da sich die Objekte in der linken Bildhälfte befinden, bleibt der rechte Ausleger in der Mittelposition, während der linke Ausleger die Ausweichbewegungen durchführt. Diese sind durch starkes Überschwingen gekennzeichnet, was auf die höhere Masse der instandgesetzten Ausleger zurückzuführen ist. Sowohl die Ausweichbewegung als auch die Positionierung in die Bildmitte schwingen stark über, wodurch das Ende eines Ausweichvorganges gleichzeitig der Startpunkt eines neuen Ausweichvorganges ist. Bei der Umfahrung des untersten Objektes ist deutlich zu sehen, dass der Frequenzumrichter nicht ordentlich auf die richtige Position regeln kann.

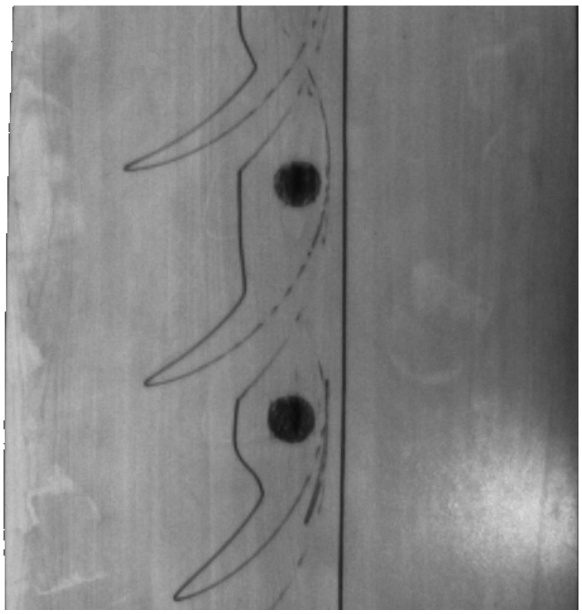


Abbildung 85: Beide Ausleger, Objekte im linken Bildbereich

Abbildung 86 zeigt die Steuerung mit beiden Auslegern wobei sich die beiden unteren Objekte in der Bildmitte befinden, sodass beide Ausleger für den Ausweichvorgang benötigt werden. Zusätzlich überschneiden sich die beiden Ausweichvorgänge, da zwischen den Objekten die Mittelposition nicht angesteuert wird. Das oberste Hindernis befindet sich nicht exakt in der Bildmitte, weshalb der rechte Ausleger eine Ausweichbewegung durchführt.

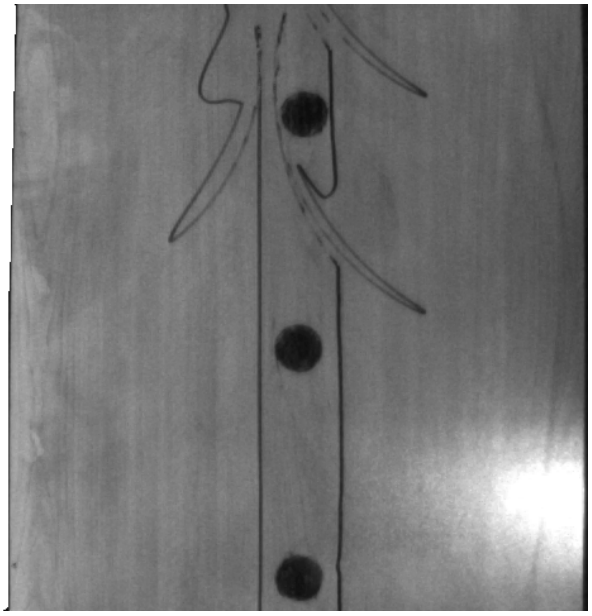


Abbildung 86: Beide Ausleger, Objekte in der Bildmitte

Der Betrieb mit beiden Servomotoren im Modus Schwingen ist in Abbildung 87 dargestellt. Die Objekte befinden sich in der Bildmitte, weshalb beide Motoren eine Ausweichbewegung durchführen. Die Position des mittleren Objektes dürfte sich etwas nach rechts verschoben haben, was passiert, wenn sich die Räder des Aufbaus beim Wechseln der Fahrtrichtung mit einiger Verzögerung in Fahrtrichtung drehen. Deshalb versucht der linke Servomotor in die Mittelposition zu erreichen. Es ist auch deutlich zu erkennen, wann die Ausholbewegung der einzelnen Frequenzumrichter erfolgt. Das Schreiben der neuen Position erfolgt zuerst für den linken Frequenzumrichter und anschließend für den rechten. Dies kann man an der Ausholbewegung für das mittlere Objekt erkennen.

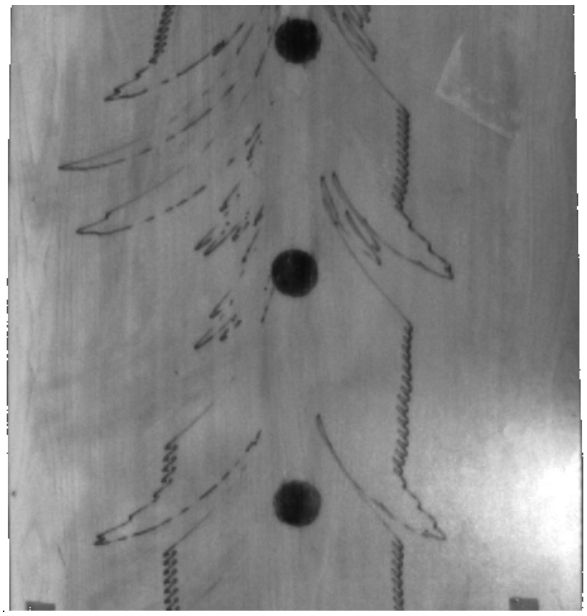


Abbildung 87: Beide Ausleger, Objekte in der Bildmitte, Betriebsmodus Schwingen

In Abbildung 88 befinden sich die Objekte im linken Bildbereich und der Betriebsmodus Schwingen wurde gewählt. Der rechte Ausleger schwingt somit in einem kleinen Bereich um

die Mittelposition, wohingegen das Schwingen bei der Ausholbewegung in einem sehr kleinen Bereich um die Ausweichposition erfolgt.

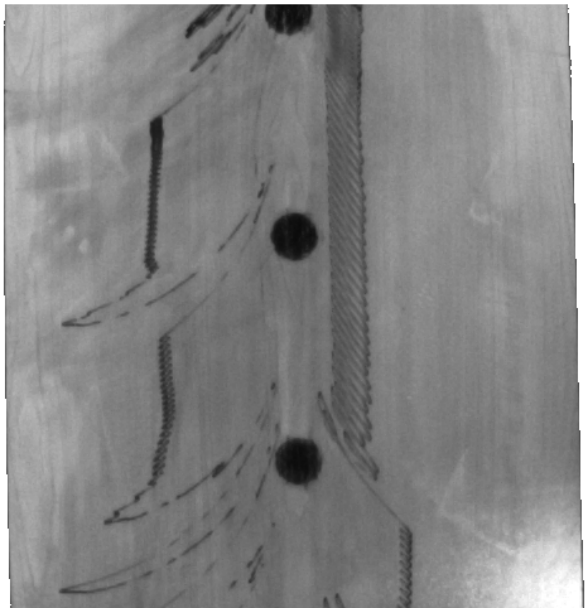


Abbildung 88: Beide Ausleger, Objekte leicht links, Betriebsmodus Schwingen

Abbildung 89 zeigt die Ausweichbewegungen für den Betriebsmodus Schwingen. Die Ausweichbewegungen starten knapp vor dem Objekt und werden nach drei Objektbreiten wieder beendet. Lediglich das starke Überschwingen am Anfang und Ende des Positioniervorganges verursacht Probleme, welche jedoch durch einen leichteren Ausleger behoben werden können.

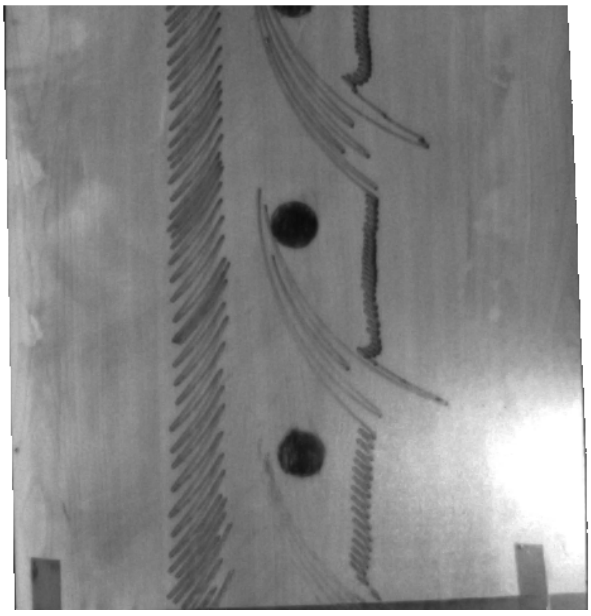


Abbildung 89: Beide Ausleger, Objekte im rechten Bildbereich, Betriebsmodus Schwingen

9. Probleme, Weiterentwicklung und Verbesserungsvorschläge

Schlechte Beleuchtungsverhältnisse können die Ergebnisse deutlich verfälschen. Beispielsweise kann eine starke Lichtquelle, die direkt auf die Holzplatte leuchtet, dazu führen, dass Objekte, die sich in ihrem Kegel befinden nicht erkannt werden können. Wird nicht die gesamte Platte beziehungsweise der unter Bildbereich stark und dauerhaft beleuchtet, so würden lediglich wenige Objekte nicht erkannt werden, wobei diese im darauf folgenden Bild möglicherweise wieder gefunden werden.
Ein weitaus größeres Problem kann Umgebungslicht und Schattenwurf darstellen. Das linke Teilbild aus Abbildung 90 zeigt das Kamerabild, in dem rechts unten auf der Holzplatte ein dunkler Fleck zu sehen ist. Durch die Segmentierung entsteht das rechte Teilbild, welches eine Vielzahl von Segmenten enthält.

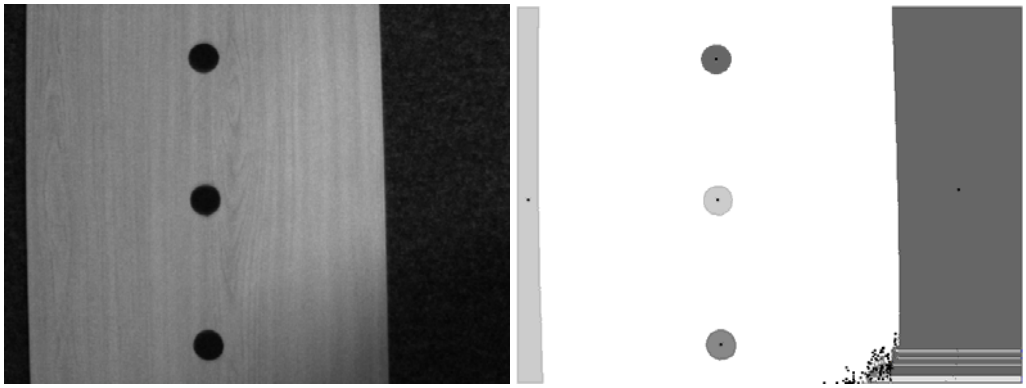


Abbildung 90: Schlechte Beleuchtung

Nach der Filterung über die Objektgröße fallen einige davon weg, jedoch nicht alle, was dazu führt, dass diese bei der Berechnung des Verschiebungsvektors berücksichtigt werden und somit ein Falscher Vektor berechnet wird. Abbildung 91 zeigt die Ergebnisbilder der Registrierung.

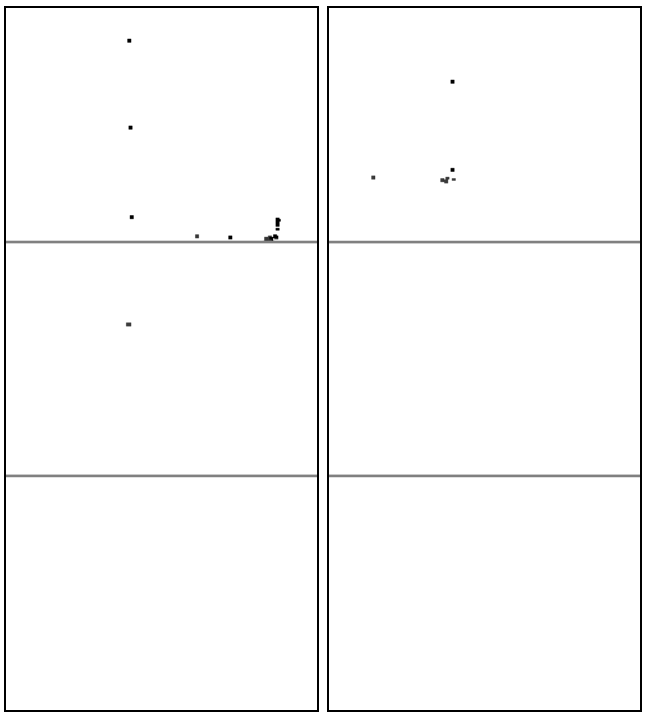


Abbildung 91: Objekte werden seitlich aus der Objektliste geschoben

Es wird ein Verschiebungsvektor mit starker x-Komponente ermittelt und so werden die Positionen der Objekte aus der Objektliste falsch berechnet. Die neuen Positionen liegen außerhalb des Listenbereichs und somit sind die Objekte aus dem rechten Teilbild verschwunden.
Dieses Problem kann auch auftreten, wenn bei direkter Sonneneinstrahlung auf dem Untergrund Flächen entstehen, die bei der Segmentierung berücksichtigt werden. Da sich diese nicht bewegen, wird ein falscher Verschiebungsvektor berechnet und somit werden die Objekte aus der Objektliste verschoben, weil ihre x-Koordinaten nicht mehr im Bildbereich liegen.
Dieses Problem kann durch eine Abschirmung des Umgebungslichtes behoben werden. Hierzu kann ein Verbau um den Sichtbereich der Kamera benutzt werden, indem eine Beleuchtung installiert wird welche den Aufnahmebereich gleichmäßig ausleuchtet.
Der Ausweichvorgang erfordert für weiter von der Bildmitte entfernte Objekte eine große Winkeländerung. Die y-Koordinate zum Starten des Ausweichvorganges liegt weit von der Motorachse entfernt. Dies erfordert einen langen Ausleger, bei dem sich die Masse am Ende des Auslegers stärker auswirkt und einen weiten Weg zur Folge hat. Ebenso sind die Koordinaten der Startzeitpunkte sehr unterschiedlich und der Abstand zwischen zwei Objekten sollte so gewählt werden, dass der Ausleger beim Überschwingen nicht mit einem anderen Objekt kollidiert. Abbildung 92 zeigt eine alternative Möglichkeit zur Montage der Servomotoren. Im linken Teil des Bildes ist die aktuelle Konfiguration dargestellt, im rechten Teil sind die Alternativen abgebildet. Die Montage der Servomotoren erfolgt zwischen der Bildmitte und dem Bildrand. In dunkelgrau wird der Schwenkbereich mit einem kurzen Ausleger dargestellt, wodurch ein Halbkreis abgefahren werden kann. Dadurch, dass die Masse näher an der Motorachse liegt als bei der aktuellen Realisierung, würde sich das Überschwingen weniger stark auswirken.
Eine andere Möglichkeit wäre die Verwendung eines etwas längeren Auslegers, in hellgrau dargestellt, der einen deutlich kleineren Ausschlagbereich besitzt. Dies hat den Vorteil, dass der Startpunkt für den Ausweichvorgang in einem wesentlich kleineren Bereich schwankt, als bei der aktuellen Realisierung und dass dadurch Objekte knapper hintereinander liegen können da die Kurve der Ausweichbewegung flacher verläuft. Jedoch muss eine Steuerung ergänzt werden, damit die Ausleger nicht miteinander kollidieren. Dies könnte auch mechanisch über einen Endanschlag gelöst werden. Da die Ausholbewegungen kürzer sind, wird die Zielposition schneller erreicht wird.

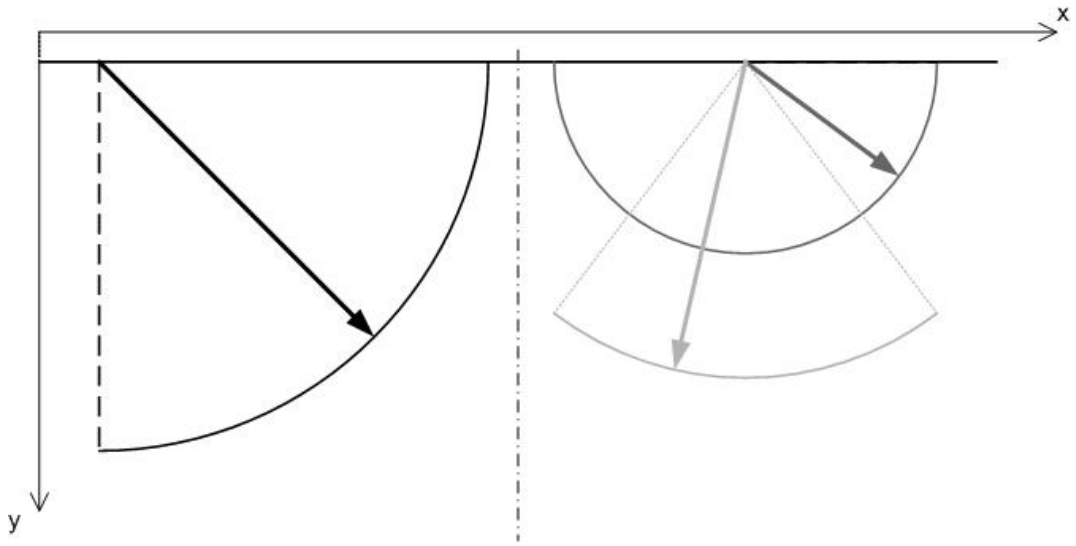


Abbildung 92: Alternative Montage der Servomotoren

Literaturverzeichnis

[1] Bernhard Baltes-Götz: Einführung in das Programmieren mit C# 2.0, Universitäts-Rechenzentrum Trier, 2007

[2] Bernhard Favre-Bulle: Automatisierung Komplexer Industrieprozesse, Springer Wien New York, 2004, ISBN 3-211-21194-2

[3] Christian Demant, Bernd Streicher-Abel, Peter Waszkewitz: Industrielle Bildverarbeitung, 2.Auflage, Springer-Verlag Berlin Heidelberg, 2002, ISBN 3-540-41977-2

[4] Fu Chang, Chun-Jen Chen, Chi-Jen Lu: A Linear-Time Component-Labeling Algorithm Using Contour Tracing Technique, Institute of Information Science, Academia Sinica, Taiwan, http://www.iis.sinica.edu.tw/~fchang/paper/component_labeling_cviu.pdf, 1.12.2008

[5] Herbert Bay, Tinne Tuytelaars, Luc Van Gool: SURF: Speeded Up Robust Features, ETH Zurich, http://www.vision.ee.ethz.ch/~surf/, 1.10.2008

[6] KEB: Combivert Betriebsanleitung Leistungsteil, http://www.keb.de/de/home.html, 07.10.2008

[7] KEB: Combicom Ethernet-Operator Version 1.1, http://www.keb.de/de/home.html, 07.10.2008

[8] KEB: Betriebsanleitung KEB Servomotore Größe A1 . . . F3, http://www.keb.de/de/home.html, 07.10.2008

[9] KEB: Entwicklungsinformation Protokollbeschreibung DIN66019II, http://www.keb.de/de/home.html, 07.10.2008

[10] Peter Haberäcker: Digitale Bildverarbeitung, 4.Auflage, Carl Hanser Verlag München Wien, 1991, ISBN 3-446-16339-5

[11] Rafael C. Gonzalez: Digital Image Processing, Pearson Education Inc. Prentice Hall, 2008, 3.Auflage, ISBN 0-13-168728-x

[12] Stefan Henn, Florian Jarre, Kristian Witsch: Mathematische Bildverarbeitung – Ein Überblick über verschiedene Modelle und Methoden zur Registrierung digitaler Bilddaten, Jahrbuch der Heinrich-Heine-Universität Düsseldorf, 2002, ISBN 3-9808514-1-9