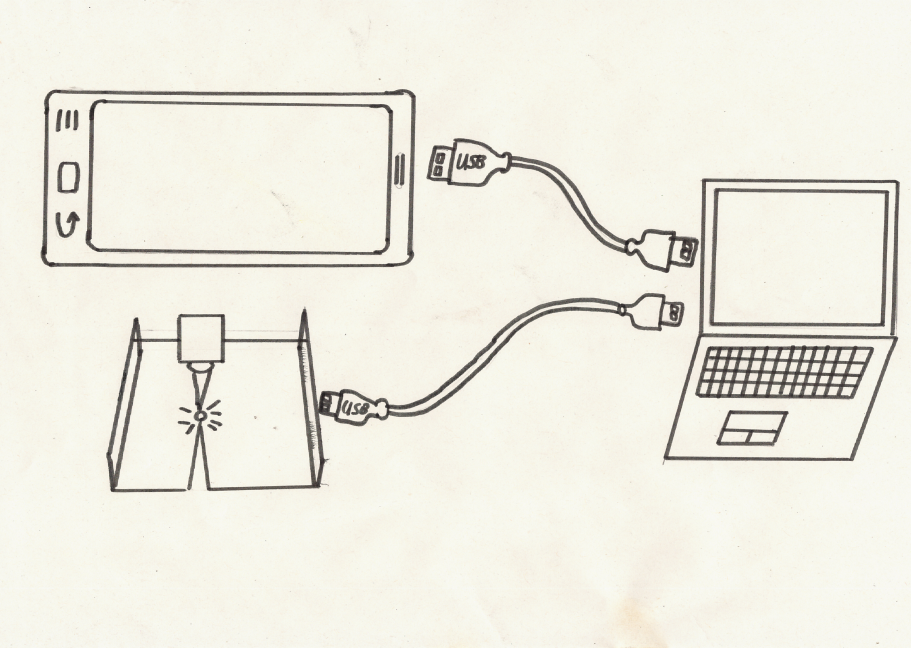

Entwicklung eines Systems zum ausschneiden von geografischer Figuren auf Basis einer fotografischen Erfassung

Bachelorarbeit

im Studiengang Technische Informatik (B.A.)

an der Technischen Hochschule Berlin

vorgelegt am: 14.04.2023

von: Robert Wodara

aus: Berlin

1. Gutachter: Dr. Edzard Höfig

2. Gutachter: Prof. Dr.-Ing. Joachim Schimkat

Technische Hochschule Berlin

Inhaltsverzeichnis

Abbildungsverzeichnis **III**

Abkürzungsverzeichnis **IV**

1 Einführung **1**
1.1 Stand der Technik . . . 2
1.2 Grundlagen der Bildrotation . . . 4
1.3 Perspektivkorrektur . . . 5
1.4 Kantenerkennung in einem Bild . . . 8
1.5 Ansteuern eines Lasercutters . . . 10

2 Konzeption **11**
2.1 Motivation . . . 11
2.2 Anforderungen an das System und dessen Konstruktion: . . . 12
2.3 Aufbau des Systems . . . 13
2.4 Ziel der Arbeit . . . 13
2.5 Planung des Systems . . . 14
2.5.1 Lösung für die Software . . . 14
2.5.2 Evolution der Rahmen Muster . . . 15
2.5.3 Entwicklung der Hardware . . . 18
2.5.4 Entwicklung des Testbildes . . . 18

3 Realisierung **19**
3.1 Vorstellung der Hard- und Software . . . 19
3.2 Der LC . . . 19
3.2.1 Ansteuern des LC . . . 21
3.2.2 Kalibrieren des LC . . . 22
3.2.3 Den LC richtig einstellen . . . 23
3.3 Der Rahmen . . . 26
3.4 Ausrichtung des Aufgenommenen Bildes . . . 27
3.4.1 Definition des Schnittmusters für den Rahmen . . . 28
3.4.2 Bildliche Erkennung der Ausgeschnittenen Kreise . . . 30
3.4.3 Bildmatrix vertikal und horizontal ausrichten . . . 36
3.4.4 Bestimmung der Genauigkeit . . . 40
3.5 Kantenerkennung im Motiv . . . 41
3.6 Generierung von CNC-G-Code aus binären Bildern . . . 42
3.7 Simulation der Ausgabe . . . 44

4 Ergebnisse **52**
4.1 Schneideergebnisse . . . 52
4.2 Einlesen und Ausrichten des Bildes . . . 52
4.3 Kanntenerkennung . . . 53
4.4 Definition der Schnittkannte . . . 54
4.5 Fazit . . . 55

5 Literatur **58**

6 Anhang **59**

Abbildungsverzeichnis

1 Motte mit Hintergrund 2
2 Motte ohne Hintergrund 2
3 Brother ScanNCut SDX1350 3
4 Kirche ohne Korrektur 6
5 Kirche mit Hilfslinien 7
6 Qualitative Unterscheidung der Kantenverarbeitungsalgorhytmen . . . 9
7 Skizze vom Aufbau des Systems 13
8 Darstellung der Rahmen in chronologischer Reihenfolge 16
9 LC Computer und offene Abdeckung 20
10 LC Computer und geschlossene Abdeckung 21
11 Werksstücke für Kalibrierung 23
12 Einstellen des LC 25
13 Vergrößerung eines Schnittmusters 26
14 Skizze vom Schnittmuster 28
15 A14 30
16 A14 Verschoben 31
17 A14 mit Kreismittelpunkten 33
18 A14 kleinster Abstand zwischen den Mittelpunkten finden 35
19 A14 mit den Punkten ABCD 36
20 ABCD als Viereck mit Sollwerten 38
21 Bildmatrix Ausgerichtet 39
22 Einfaches Kontrastreiches Motiv 41
23 Motiv nach Kantenerkennung 42
24 Original und generierter G-Code im Vergleich 49
25 Original und generierter G-Code im finalen Vergleich 51
26 Überprüfung der Schnittkante 54

Abkürzungsverzeichnis

LC Lasercutter

SAG:FArBE Ausschneiden von geometrischen Figuren auf Basis einer fotografischen Erfassung

1 Einführung

Diese Bachelorarbeit behandelt den Aufbau eines Systems, das auf Grundlage einer fotografischen Erkennung geometrische Figuren selbstständig ausschneiden kann. Das hier entwickelte System soll dabei helfen, Motive auf analogen Bildern von ihrem Hintergrund mit einer möglichst hohen Genauigkeit zu trennen. Dabei liegt der Fokus dieser Arbeit auf der Konstruktion dieses Systems und auf der Lösung der technischen Herausforderungen, die die Problematik mit sich führt. Ein spezielles Augenmerk wird hier auf die Koordinatentransformation der fotografischen Erfassung und des analogen Bildes gelegt. Ein weiteres Gebiet, das hier behandelt wird, ist das korrekte Auslesen der Bilddaten und die präzise Weitergabe an das Schneidewerkzeug, um einen möglichst sauberen Schnitt zwischen dem Motiv und seinem Hintergrund zu erzeugen. In der Einführung werde ich den aktuellen Stand der Technik darstellen und einen Bezug meiner Arbeit zu gängigen Praktiken für die Lösung der auftretenden Probleme aufzeigen.

1.1 Stand der Technik

Es ist heutzutage mit nur wenig Aufwand und praktisch keinen Vorkenntnissen möglich, das Motiv eines digitalen Bildes von seinem Hintergrund zu trennen. Eine kostenlose und einfache Umsetzung bietet zum Beispiel die Webseite: www.picwish.com. Hier kann man kostenlos ein beliebiges Bild hochladen und die Webseite sucht selbstständig das Motiv heraus und bietet einen kostenlosen Download des ausgeschnittenen Bildes an. Ein Beispiel dafür zeigen die folgenden Abbildungen:

Abbildung 1: Motte mit Hintergrund

Abbildung 2: Motte ohne Hintergrund

Wünscht man eine höhere Qualität beim Trennen von Motiv und Hintergrund, so sollte man sich mit gängigen Programmen der Bildbearbeitung beschäftigen, wie z.B Photoshop. Das hier gezeigte Beispiel ist nur eines, welches einen sehr geringen Aufwand

in Form von Zeit und Geld verbraucht.

Geräte, die dem hier vorgestellten System am nächsten kommen heißen SSchneideplotter". Eingescannte oder vorher definierte Motive kann ein Schneideplotter in ein etwa 3 mm dickes Material einschneiden. Ein Beispiel für einen Schneideplotter ist der "Brother ScanNCut SDX1350 Schneideplotter"
[1].

Abbildung 3: Brother ScanNCut SDX1350

Diese Geräte haben die Eigenschaft, dass sie nie das Originalbild bearbeiten, sondern immer eine Kopie des Bildes herstellen und einen Rahmen aus einem gewählten Material ausschneiden. Dies soll das hier vorgestellte System auch können. Die primäre Funktion dieses Systems soll aber sein, das Originalbild mit einer höheren Präzision zu schneiden, als es mit der Hand möglich ist.

Die hier verwendete Mathematik [1] sowie der Programmcode basieren auf Algorithmen, die schon sehr lange Zeit bekannt sind.

[1]Foto und Daten von: https://www.naehwelt-flach.de/brother-scanncut-sdx1350-schneideplotter.html?gclid=Cj0KCQiAx6ugBhCcARIsAGNmMbhw–1rnzjs8h7mhhhAI4L16NIYh1LlqjVzFvSjYy7Aa$_R JWwdLFu8aAhBDEALw_w cB$

1.2 Grundlagen der Bildrotation

In dieser Arbeit wird verstärkt darauf eingegangen, dass ein Bild rotiert werden muss. Die theoretischen Grundlagen finden Sie im Buch "Computer Vision" [2].

Zitat Anfang: "Eine Rotation kann .. im Zweidimensonalen .. durch eine Matrixmultiplikation ausgedrückt werden. Gegeben sei ein Vektor χ . Wird er als ... Ortsvektor interpretiert, so wird die Drehung des Punktes um den Ursprung des Koordinatensystems berechnet.
Im R^2 ist die Berechnung einer solchen Rotationsmatrix eindeutig, da es nur eine Drehphase existiert. Gegeben sei ein Vektor $\chi = (x,)$ und ein Drehwinkel Θ. Die Drehung gegen den Uhrzeigersinn von χ um den Winkel Θ berechnet sich zu:
$\begin{pmatrix} \frac{x'}{y'} \end{pmatrix} = \begin{pmatrix} \frac{cos\theta * -sin\theta}{sin\theta * cos\theta} \end{pmatrix} * \begin{pmatrix} \frac{x}{y} \end{pmatrix}$

Im R existieren drei Basisrotationen um die Achsen x, y und z.

$$R_x(\Theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\Theta & -sin\Theta \\ 0 & sin\Theta & cos\Theta \end{pmatrix}$$

$$R_y(\Theta) = \begin{pmatrix} cos\Theta & 0 & sin\Theta \\ 0 & 1 & 0 \\ -sin\Theta & 0 & cos\Theta \end{pmatrix}$$

$$R_z(\Theta) = \begin{pmatrix} cos\Theta & -sin\Theta & 0 \\ sin\Theta & cos\Theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Aus diesen Rotationen kann nach Euler's Theorem mit drei Variablen jede beliebige Rotation im Raum zusammengesetzt werden. Hierzu existieren zwei verschiedene Konventionen für die Interpretation der Reihenfolge der Einzelrotationen. Für $raumfeste$ Drehachsen werden die Einzelrotationen von rechts nach links interpretiert, wie anhand des folgenden Beispiels zu sehen ist:
$R_{XYZ}(\alpha\beta\gamma) = R_Z(\gamma) R_Y(\beta) R_X(\alpha)$ " Zitat ende [3].

1.3 Perspektivkorrektur

In diesem Kapitel werde ich die Grundlagen und die Notwendigkeit der Perspektivkorrektur erläutern. Ich stütze mich hierbei auf einen Blogbeitrag von "www.bonnescape.info" wo die Problematik sehr anschaulich dargestellt und erklärt wird:

"Natürlich sehen wir alles in unserer Nähe relativ groß und weit Entferntes entsprechend kleiner. Daraus ergibt sich eine Perspektive, die beispielsweise bei einem von unten betrachteten hohen Gebäude eine Fassade so abbildet, dass sie sich nach oben zu verjüngen scheint. Wir denken nicht weiter über diesen Sachverhalt nach, sind aber geprägt durch diese Sehgewohnheit, obwohl wir selbstverständlich wissen, dass die sich verjüngenden ßtürzenden"Kanten der Fassade mit großer Wahrscheinlichkeit parallel und senkrecht verlaufen. Entsprechend abgebildet werden sie bei der Aufnahme eines Fotos aber nur dann, wenn sie sich in einer Ebene befinden, die parallel zum Film oder zum Sensor in der Kamera verläuft. Dies kann mit einer Fachkamera oder einem Shift-Objektiv gewährleistet werden, indem die Kamera parallel zum Objekt ausgerichtet wird und in dieser Konstellation der Bildausschnitt verschoben wird. Hat man keine solche Ausrüstung zur Hand, kann das Bild auch später am Rechner durch Verzerren so manipuliert werden, dass ein ähnlicher Effekt entsteht. [...]

Abbildung 4: Kirche ohne Korrektur

Ein Foto der griechisch-orthodoxen Kirche von Oia auf Santorini ohne Perspektivkorrektur. Die Kamera war leicht nach oben gerichtet, um den Kirchturm vollständig ins Bild zu bekommen. Die Aufnahme erfolgte mit einem 35mm-Objektiv.
Da wir gewohnt sind, Fotos von hohen Türmen in dieser Weise zu sehen, erscheint die Abbildung zunächst nicht unnatürlich. Den Architekturfotografen stört jedoch zumindest die schief stehende Säule unten links und die Tatsache, dass der Glockenturm nach hinten wegzukippen scheint.

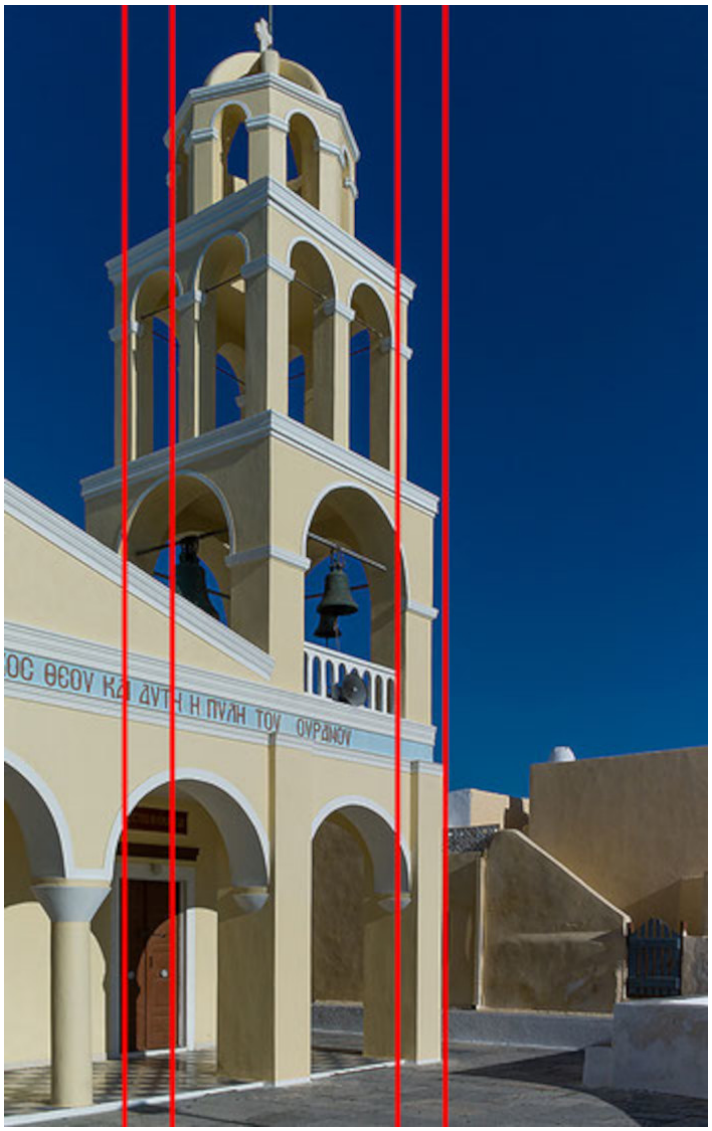

Abbildung 5: Kirche mit Hilfslinien

Bei dieser Variante wurde eine Korrektur bis zur Parallelstellung der vertikalen Linien vorgenommen. Tatsächlich sieht man das jedoch nur bei der Portalsäule unten links. Der Kirchturm vermittelt dagegen den Eindruck, dass er oben auseinandergeht und scheint proportional viel zu groß für den Rest des Gebäudes [...]

Das Erzwingen der Parallelität vertikaler Kanten im Foto scheint gerechtfertigt, weil wir ja wissen, dass sie parallel verlaufen, bewirkt aber tatsächlich eine Verfremdung des Objektes, weil sie unseren Sehgewohnheiten widerspricht." [4].

1.4 Kantenerkennung in einem Bild

Die Technik der Kantenerkennung ist für diese Arbeit ein wichtiger Baustein. Daher soll die dahinterstehende Mathematik in diesem Kapitel erläutert werden. Dabei beziehe ich mich auf das Kapitel 11: 'Kantendetektion' aus dem Buch 'Bildverarbeitung in der Praxis': "Versuche haben gezeigt, dass Menschen sich beim Betrachten von Objekten sehr stark auf die Grenzen zwischen mehr oder minder homogenen Regionen konzentrieren. Gegenstände werden meist schon anhand der groben Umrisse erkannt. Auch bei der digitalen Bildverarbeitung stellen diese Kanten bzw. Konturen eine wichtige Stufe zur Bildsegmentierung, Objekterkennung und somit zur Bildinterpretation da. Wie gut diese Kanten in den Vorlagen gefunden werden, ist abhängig von der Qualität des Bildes, der Art der Kanten und den verwendeten Algorithmus. Je nach Kantenart eignet sich das eine oder andere Verfahren besser". [5]

Die Kantendetektionsverfahren lassen sich grob ïn zwei Klassen unterteilen, die der Parallelenverfahren und die der sequenziellen Verfahren. Die Parallelenverfahren stellen auch oft eine Vorstufe für die Sequenziellen da. Bei den parallelen Verfahren wird lokal ein Eigenschaftsvektor bestimmt, der Angaben wie Kantenstärke, Kantenrichtung oder Maße für die Kantenform enthält. Da dieser Eigenschaftsvektor nur von lokalen bet-Funktionen abhängt, kann er parallel für alle anderen Bildpunkte berechnet werden. Je nach Verfahren werden die einzelnen Eigenschaften als Kriterium für das Vorliegen eines Kantenpunktes verwendet". [6]

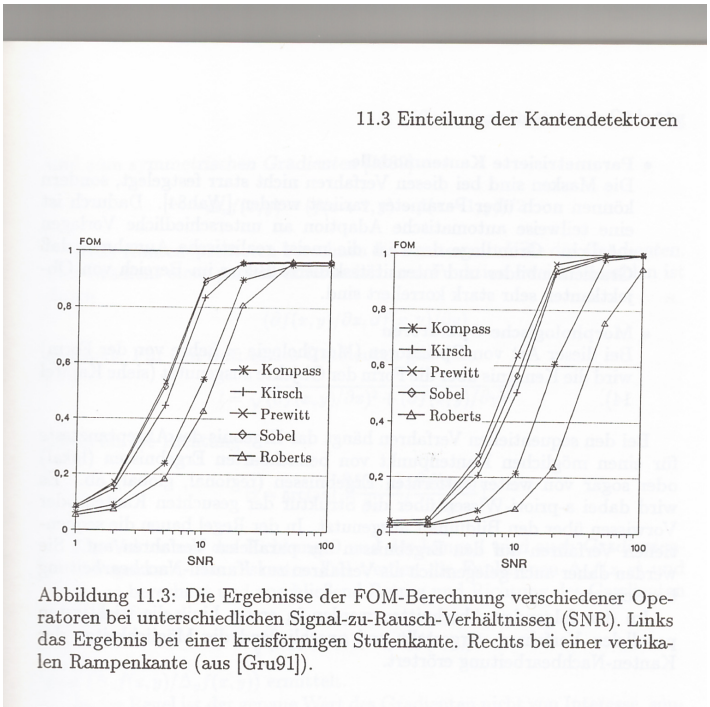

Abbildung 6: Qualitative Unterscheidung der Kantenverarbeitungsalgorhytmen

Das grundliegende Prinzip einer Kantenerkennung basiert auf den Vergleich eines lokalen Ausschnitts von einem Bild mit einer Maske. Die Definition der Maske, sowie die vergleichenden Operatoren sind die entscheidenden Merkmale, bei denen sich die verschiedenen Algorithmen unterscheiden.
Wie die Grafik schon vermuten lässt, gibt es verschiedene Algorithmen zur Kantenerkennung.
"Der einfache Differenzoperator ist die direkte Umsetzung des Rückwärtsgradienten in eine Filter-Maske und hat folgende Gestalt:

$$\Delta_x f(x,y) = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \Delta_x f(x,y) = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Die Maske Δ_x spricht vor allem auf vertikale, Δ_y auf horizontale Kanten an. Die Berechnung ist einfach, da nur eine 2 x 2-Matrix zu betrachten ist [...] und nur Additionen verwendet werden [...]". [6]

1.5 Ansteuern eines Lasercutters

Ein LC ist die Basis meiner Arbeit und der prinzipielle Umgang mit diesem Gerät soll hier erläutert werden. Es gibt LC in vielen verschiedenen Variationen. Hier wird nur ein Modell beschrieben, welches Ähnlichkeit mit dem für diese Arbeit verwendeten LC aufweist. Meine Beschreibungen dieser Maschine basieren alle auf den Buch 'Lasercutting' [7]. Eine treffende Formulierung, die in dem Buch auf Seite 9, Absatz 1 steht ist, dass der Lasercutter die "digitale Laubsägeïst. Die Funktion, die der Lasercutter hier in der Arbeit erfüllen soll, wird durch diese Formulierung gut beschrieben. Sie suggeriert allerdings auch einen gewissen Mangel an Präzision, der nicht vorliegt. LC sind unverhältnismäßig präziser als eine von Hand geführte Säge, wenn wir von einer Jahrzehnte geübten Hand mal absehen. Es ist ein Schneidegerät, das mit Hilfe eines stark gebündelten Lichtstrahls eine so große thermische Energie transportiert, dass das durch den Laser beleuchtete Material verbrennt, verkohlt oder schmilzt. Natürlich muss das Material in ihrer Beschaffenheit und Dicke der schneidenden Maschine angepasst sein.

Angesteuert werden diese Maschinen unter anderem mittels eines 'G-Codes'. Dies ist eine alte "Programmiersprache für nummernbasierte Steuerung" [8] aus den 1950er Jahren, die eine sehr einfache Bedienung der Maschine erlaubt.

2 Konzeption

In dem nun folgenden Kapitel möchte ich das von mir aufgebaute System beschreiben und die von mir getroffenen Entscheidungen darstellen und erläutern.

2.1 Motivation

Ich fotografiere gerne mit einer analogen Kamera. Nicht jedes Bild wird so, wie ich es mir wünsche und so kommt die eine oder andere Entwicklung in den 'Bastelkarton'. Dort liegt dann ein Stapel ungenutzter Bilder, die ich gerne zu neuen Bildern zusammenstelle, wenn da nicht der aufwendige Zwischenschritt wäre, dass ich das Motiv von dem Bild mittels einer Schere oder eines Skalpells trennen müsste. Dies erfordert große Präzision und Geduld, sodass eine Collage von drei Motiven schnell zwei Stunden dauern kann. Diesen Prozess des Ausschneidens soll nun das **S**ystem zum **A**usschneiden von **g**eometrischen **F**iguren **a**uf **B**asis einer fotografischen **E**rfassung. Oder kurz: SAG:FArBE übernehmen. Das System besteht aus drei Haupt-Hardwarekomponenten: Einer Kamera, einem Computer mit Bildschirm, Maus und Tastatur und einem LC (Lasercutter).
Eine Software verbindet die drei Komponenten. Die Kamera ist so angebracht, dass sie die Arbeitsfläche des Laserschneiders einsehen kann. Auf der Arbeitsfläche befindet sich eine Konstruktion, die das zu zerschneidende Bild fixiert. Ebenfalls auf der Arbeitsfläche befindet sich eine grafische Orientierungshilfe, die dem System ein präzises Arbeiten ermöglicht. Die Bilder dürfen zwei verschiedene Formate haben: 10 x 15 cm und 9 x 13 cm. Nachdem das zu zerschneidende Bild fixiert wurde, wird ein Foto von der Arbeitsfläche gemacht. Das Bild wird von der Software von SAG:FArBE verarbeitet. Die Software nimmt das Bild auf, erlaubt der anwendenden Person Schnittlinien zu definieren und übermittelt diese Schnittlinien an den Laserschneider, welcher das Bild dann mit einer definierten Präzision ausschneidet. Nachdem alle Schnittlinien benannt wurden, erstellt die Software einen Ablaufplan für den LC. Das Ergebnis soll sein, dass die durch den User definierten Schnittlinien sauber auf das Bild übertragen werden.

2.2 Anforderungen an das System und dessen Konstruktion:

1. Der LC wird mit einer definierten Genauigkeit hergestellt und geliefert. Diese Genauigkeit gilt es zu überprüfen und für das Arbeiten mit den Bildern zu erhalten. Mit der Hand kann ich ein Motiv auf 1 mm genau ausschneiden. Das SAG:FArBE sollte eine Absolute Abweichung von 0,5 mm nicht überschreiten.

2. Um die Präzision des Systems qualitativ messen zu können, muss ein Testbild in beiden oben genannten Formaten erstellt werden, welches die Grenzen der Genauigkeit unterfordert, erreicht und überschreitet.

3. Das absolute Maß von 0,5 mm ist die Richtlinie, nach der sich die Konstruktion des Systems richten soll. Die Hardware sollte imstande sein, diese Richtlinie einzuhalten. Eine Berechnung der Genauigkeit der Kamera muss ebenso vorliegen wie ein Computer, der mit der Größe der Bilder umgehen kann.

4. Das SAG:FArBE soll ein Kalibrierungsprogramm haben, welches für die bedienende Person leicht zu benutzen ist.

5. Die Software beinhaltet eine grafische Oberfläche, mit der das System bedient werden soll.

2.3 Aufbau des Systems

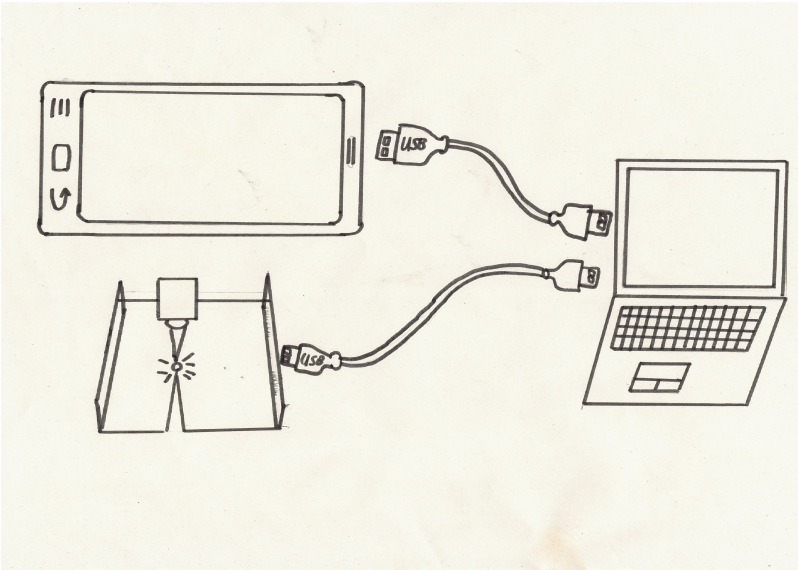

Abbildung 7: Skizze vom Aufbau des Systems

Die Skizze zeigt die drei Hauptkomponenten, die ich für die zugrunde liegende Arbeit verwenden werde. Der LC, das Smartphone und der Computer sind alle drei über ein USB-Kabel miteinander verbunden. Der LC sowie der Computer liegen auf einem Tisch. Der LC braucht eine entsprechende Schutzumgebung, die ihn zum einen ein sicheres Arbeiten ermöglicht und zum anderen die anwendende Person vor zu hoher Lichtbelastung schützt. Das Smartphone wird über den LC gehalten und übernimmt die Aufgabe des Aufnehmens eines Bildes. Dieses Bild wird an den Computer übermittelt und dieser berechnet aus den ihm gelieferten Informationen die Bewegungsabläufe, welche der LC nun zu fahren hat.

2.4 Ziel der Arbeit

Ziel dieser Bachelorarbeit ist es, ein Gerät zu entwickeln, das geometrische Figuren ausschneidet. Diese geometrischen Figuren befinden sich auf Papier welches vorher auf die Schnittarbeitsfläche fixiert wurde. Das Gerät soll in X und Y-Richtung eine maximale Abweichung von 0,5 mm erreichen. Die Prämisse, die ich mir beim Ausarbeiten der Lösung gestellt habe, lautet: ”keep it simple and smart”. Ich habe immer nach einem noch einfacheren Weg gesucht, die Teilprobleme zu lösen, sodass ich am Ende eine leicht

verständliche und zugängliche Arbeit abgeben kann.

2.5 Planung des Systems

In Kapitel möchte ich meine Planung für das System und dessen Entwicklung darstellen.

2.5.1 Lösung für die Software

Zu Beginn wurde die Software für dieses System auf der Android Plattform entwickelt. Da jedes moderne Smartphone eine Kamera, viel Rechenleistung und eine einfache Bedienung verspricht, war der Gedanke naheliegend, das System für den 'Computer in der Hosentasche' zu entwickeln. Im Einzelnen sollte die Software folgende Aufgaben erfüllen:

- Die Software sollte im Stande sein, eine Aufnahme der Arbeitsfläche und dem auszuschneidenden Motiv in ausreichend guter Qualität zu erstellen.
- Die Aufnahme soll Hilfslinien für eine Perspektivkorrektur beinhalten und diese muss von der Software durchgeführt werden.
- Die Software soll eine möglichst angepasste Menge an Schnittlinien anbieten, welche von der benutzenden Person ausgewählt werden können.
- Die Software soll zum Lasercutter eine Verbindung aufbauen und ihm den richtigen Bewegungsablaufplan zur Verfügung stellen.

Für jedes hier dargestellte Teilproblem fand sich eine App oder eine Klasse, die ich für diese Arbeit verwenden konnte. Besonders bei den gefundenen Apps hat es sich aber gezeigt, dass sie nicht nur meine gewünschte Funktion beinhalteten, sondern auch eine große Menge häufig schlecht dokumentierter anderer Funktionen, die mühevoll von dem zu trennen waren, was für diese Arbeit eigentlich nützlich war. Die zweite Herausforderung bei der Nutzung der Android Plattform war, dass die nun sauberen Methoden und Klassen wieder miteinander verbunden werden mussten. Diese beiden Teilaufgaben erschienen mir als zu aufwendig und ich wandte mich von der Android Plattform ab.
Den nächsten Lösungsansatz suchte ich in der Programmiersprache 'C'. Durch die vorangegangene Suche für die Android Plattform waren meine Kriterien nun für die

ausgewählten Programme zur Teillösung meiner Probleme strenger. Ich legte besonders großen Wert auf eine verständliche und umfangreiche Dokumentation der gefundenen Arbeiten. Diese Kriterien wurden durch viele Programme nicht erfüllt, sodass ich ein weiteres Mal die Programmiersprache wechseln wollte.
Ich entschied mich für eine Programmiersprache, die von vornherein auf die Bearbeitung von Matrizen ausgelegt war. Da hinter der neuen Programmiersprache ein Unternehmen steht, das ein wirtschaftlichen Interesse daran hat, dass möglichst viele Menschen es nutzen, sind sehr viele Programmbeispiele sehr ausführlich und gut dokumentiert. Innerhalb kürzester Zeit fand ich Funktionen, die meine Aufgaben lösen konnten. In letzter Konsequenz entschied ich mich daher für MatLab als Programmierumgebung.

2.5.2 Evolution der Rahmen Muster

Als Rahmen wird im Kontext dieser Arbeit ein Gegenstand bezeichnet, der folgende Eigenschaften aufweist:
Der Rahmen...

1. ... sollte schnell und einfach reproduzierbar sein
2. ... sollte den Schneidevorgang nicht behindern
3. ... sollte die Möglichkeit bieten, auf eine schnelle und einfache Art und Weise die Position des Bildes relativ zur Position des LC zu definieren

Die Summe an Anforderungen zu erfüllen, erwies sich als eine Aufgabe, die in evolutionären Schritten bewerkstelligt werden musste.

Abbildung 8: Darstellung der Rahmen in chronologischer Reihenfolge

Das Bild, "Darstellung der Rahmen in chronologischer Reihenfolge" zeigt die Entwicklung der Rahmen in chronologischer Reihenfolge, wobei die oberste Layer die älteste ist. Es sind 5 Schnittmuster dargestellt und im fortlaufenden Text werden sie nach ihrem Entwicklungsstand benannt, wobei die oberste Layer Layer 1 ist und die unterste Layer Layer 5. Im fortlaufenden Text werden nun die Erkenntnisse aus der vorherigen Layer erläutert und mit der aktuellen Layer verglichen.

Vor der Layer 1 gab es die Idee, einen Rahmen mittels 3D Druck zu erzeugen, der eine reale Begrenzung für die auszuschneidenen Bilder bieten würde. Diese Idee hat sich aber nicht durchgesetzt, da sie die Anforderungen 1 und 2 nicht erfüllen könnte.

Layer 1 hat, wie alle Layer an den vier Eckpunkten eine Markierung, die für die Bildverarbeitung als Erkennungspunkt dienen sollte. Wie bei allen Layern sind die Abstände zwischen den Markierungen 150 mm entfernt, was der maximalen Bildbreite entspricht. Gefüllt ist dieses Quadrat mit horizontalen und vertikalen Linien, die es ermöglichen sollten, das Bild möglichst gerade hinzulegen. Layer 1 besteht aus einem Verpackungskarton mit ca. 0,6 mm Dicke und die Linien wurden mit dem Lasercutter mit einer Geschwin-

digkeit von 2000 mm/ min und einer Leistung von 44 % in zwei Durchläufen eingebrannt.

Die Nachteile von Layer 1 sind, dass die Erkennungspunkte sehr klein und wenig kontrastreich sind. So sind sie bei nicht optimalen Lichtverhältnissen schwer zu erkennen. Ein weiterer Nachteil ist, dass die Hilfslinien zwar dafür sorgen, dass man das Bild relativ zum Rahmen gerade legen kann, dieser Effekt aber vollkommen hinfällig ist, da auf einem fotografierten Bild selten gerade Kanten zu sehen sind. Somit erzeugen die Hilfslinien nur überflüssige Informationen auf dem aufgenommenen Bild, welche wieder herausgefiltert werden müssen.
Die Idee für Layer 2 bestand darin, die Erkennungspunkte groß und kontrastreich zu machen. Hinzu kommt, dass auf die Hilfslinien verzichtet wurde. Layer 2 wurde auf eine weiße 3 mm starke, laminierte, mitteldichte Holzfaserplatte (MDF) gebrannt.

Der Nachteil von Layer 2 ist, dass der Kontrast zwischen dem eingebrannten 'Schwarz' und dem sauberen 'Weiß' nicht so groß war wie erhofft. Außerdem dauerte das Färben der schwarzen Elemente knapp über 30 Minuten, was wiederum den Kriterienpunkt 1 nicht erfüllt. Hinzu kommt, dass das Verbrennen der Laminatschicht mindestens stinkende bis gesundheitsgefährdende Dämpfe freisetzt, was ein häufiges Wiederholen der Herstellung des Rahmens unangenehm gestaltet. Daher wurde das neue, große und kontrastreiche Schnittmuster von Layer 2 auf Layer 3 übertragen mit der Veränderung, dass die schwarzen Elemente des Rahmens nicht mehr eingebrannt, sondern ausgeschnitten wurden und die losen kleinen Teile händisch abgetragen wurden. Der hohe Kontrast entstand nun durch die raue Seite einer Siebdruckplatte, welche unter dem Blatt Papier lag.

Somit war Layer 3 ein schnell und günstig herzustellender Rahmen, welcher den Schneidevorgang nicht behinderte. Die 1/4-Kreise, die als Eck-Markierungen dienten, waren aber ein grafisch komplexes Gebilde, das nicht ohne Anwendung komplexer Module erkannt werden konnte. Also wurde die geometrische Figur des 1/4-Kreises linearisiert, sodass zwei rechtwinklige Dreiecke, deren rechte Winkel aufeinander saßen, entstanden. Somit sind wir bei der Layer 4 angekommen.

Doch auch die relativ einfachen Dreiecke konnten noch vereinfacht werden und schnell

wurden aus zwei Dreiecken zwei Kreise. Der Kreis ist von Seiten der Programmierung leicht zu erkennen. Die endgültige Layer 5 wurde definiert und wird im Kapitel: 'Ausrichten des Aufgenommenen Bildes' näher beschrieben.

2.5.3 Entwicklung der Hardware

Die Hardware ist nachvollziehbarerweise das Element, das am schwierigsten und kostenintensivsten zu wechseln wäre. Für die Entscheidung der Anschaffung des Lasercutters habe ich mir zwei Monate Zeit gelassen. Nachdem ich aber ein Gerät gefunden habe welches theoretisch alle Anforderungen für dieses und eventuell künftige Projekte erfüllt, und die von mir gewünschten Kriterien auch nachgemessen wurden, gab es keine Änderung mehr im diesem Bereich.

2.5.4 Entwicklung des Testbildes

Die Entwicklung eines Testbildes ist in diesem Projekt nur in zwei Stufen zu unterteilen. Die erste Stufe sah vor, zwei Testbilder in unterschiedlichen Dimensionen anzulegen. Die zweite Stufe reduziert die Anzahl der Testbilder auf eines und erweitert dieses auf die maximale Größe, welches von den zu bearbeitenden Bildern geliefert werden könnte: Es ist 150 x 150 mm groß.

3 Realisierung

In den nun folgenden Kapitel werde ich meine Lösung der Probleme beschreiben und einen detaillierten Einblick in die Herangehensweise liefern.

3.1 Vorstellung der Hard- und Software

- Der LC wurde von mir mit großer Sorgfalt und Bedacht ausgewählt. Ich entschied mich für den 'S9' von **SCULPFUN**. Laut Hersteller hat das Gerät eine Genauigkeit von 0,01 mm und zudem einen großzügig dimensionierten Arbeitsbereich von etwa 450 x 450 mm und einen Laser mit einer Festbrennweite von 20 mm. Die effektive Lichtleistung von 5,5 Watt des Lasers erschien mir ausreichend und Tests haben gezeigt, dass er den Anforderungen für dieses Projekt genügt.
- Ich nutze die Kamera meines Smartphones. Ein **Samsung Note S8** mit einer 12 Megapixel Kamera auf der Rückseite.
- Der Computer ist ein **Lenovo E540**.
- Die primäre Programmierumgebung ist **MatLab** in der Version R2022b.
- Zum Erstellen von G-Code und Schnittmuster nutze ich die Software **LIGHTBURN** mit der Version 1.3.01.
- Das Testbild wurde mit **FreeCAD** mit der Version: 0.20.2 erstellt.

3.2 Der LC

Um einen sicheren Umgang mit dem LC zu gewährleisten, benötigt man eine Arbeitsunterlage, die auch bei hoher Hitzeeinwirkung schwer entflammbar und nach Möglichkeit von dem LC nicht zu zerschneiden ist. Ich habe mich für eine 30 mm dicke Küchenarbeitsplatte entschieden. Sie bietet physische Festigkeit und ist schwer entflammbar. Tests haben gezeigt dass der LC im Stande ist, etwa 17 mm dickes Kiefernholz zu zerschneiden. Eine annähernd Verdopplung dieser Dicke gibt mir das sichere Gefühl mit dem LC arbeiten zu können. Desweiteren fügte ich auf dem Arbeitsbereich des LCs Begrenzungen ein, in die der LC eingefasst wird und sich somit in keine Richtung bewegen kann. Ein zusätzliches Festschrauben des LCs sorgt dafür,

dass sich dieser während der Arbeit nicht verstellen kann. Auf einen Rahmen mit einer Seitenhöhe von etwa 28 cm befestigte ich eine 3 mm Pressholzplatte als Dach und als eine Seitenwand, um einen Lichtschutz bei der Arbeit zu gewährleisten. Die drei übrigen Seitenwände können bei mir frei bleiben, da sie von anderen Möbeln verdeckt werden. Ein Aluminiumprofil von etwa 2 mm Dicke, 5 cm Breite und 55 cm Länge, welches parallel zur y-Achse auf die Arbeitsfläche des LCs mittels zwei Gewinden und Flügelmuttern geschraubt wird, sorgt dafür, dass das zu bearbeitende Werkstück sicher fixiert werden kann.

Mit dieser Konstruktion ist es möglich den LC in einer Wohnumgebung arbeiten zu lassen, ohne ständig die vom Hersteller empfohlene Schutzbrille zu tragen. Ein Loch in der Dachplatte von 28 mm Durchmesser erlaubt es aus einer definierten Position heraus eine Aufnahme von der Schnittarbeitsfläche zu erstellen. Auf eine gute Belüftung des Raumes ist zu achten.

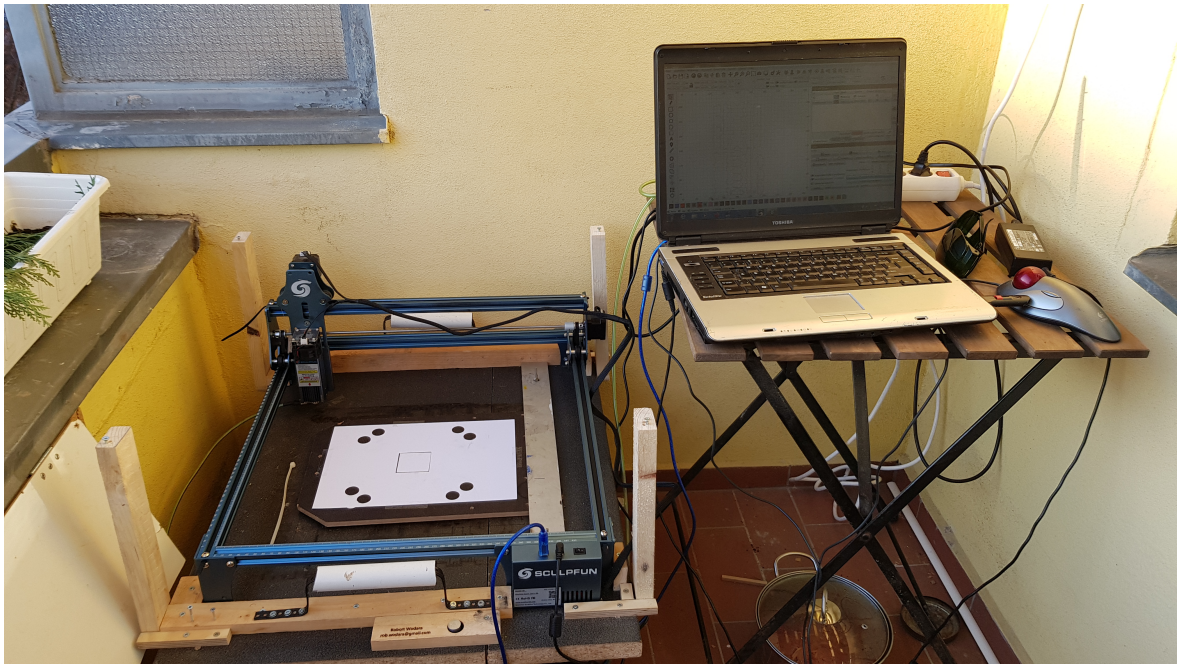

Abbildung 9: LC Computer und offene Abdeckung

Abbildung 10: LC Computer und geschlossene Abdeckung

3.2.1 Ansteuern des LC

Der LC lässt sich verhältnismäßig einfach über eine serielle Schnittstelle über einen 'G-Code' ansteuern. Der 'G-Code' ist eine überwiegend vereinheitliche Maschinensprache für Maschinen mit NC-Steuerung [7].

```
;% Robert Wodara test01
;% GRBL device profile, current position
;% https://www.maschinfo.de/CNC-G-Code
G00 ;% Positionierung im Eilgang
G17 ;% Auswahl Ebene XY
G40 ;% Fräserradiuskorrektur aus
G21 ;% Einheiten in mm
G54 ;% Koordinatensystemverschiebung 1
;G91 ;% relative Postionsangaben
G90 ;% Absolute Postionsangaben
M4  ;% Spindel gegen den Uhrzeigersinn
M8  ;% Kühlmittel 2 an
M3  ;% Spindel im Uhrzeigersinn
G0 X230.00Y-190.000
```

```
15 G1 X250.00Y-190.00S2F1000 ;% Leistung 0,2% Geschwindigkeit 1000 mm/min
16 G1 X250.00Y-220.00
17 G1 X230.00Y-220.00
18 G1 X230.00Y-190.00
19 G0 X0.00Y0.00
20 M9
21 ;G1 S0
22 M5
```

Quellcode 1: Einlesen und Verschieben von einem Bild

Der hier dargestellte G-Code dient als anschauliches Beispiel für diese Steuerungsart. In Zeile 10 wird mit dem Kommando G90 die absolute Positionsangabe ausgewählt. In Zeile 14 wird der Laserkopf auf die Position 230 mm in Richtung x und - 190 mm in Richtung y bewegt. Dies kommt daher, dass meine gewählte Startposition immer die linke obere Ecke ist und der Lasercutter sich mit diesem Befehl ungefähr in die Mitte der Arbeitsfläche bewegt. Die G1 Befehle weisen die Maschine an, sich mit der in Zeile 15 definierten Geschwindigkeit und in der gleichen Zeile definierten Leistung des Lasercutters zu bewegen. In diesem Fall ist es eine Leistung von 0,2% und einer Geschwindigkeit von 1000 mm in der Minute. Die Zeilen 15 bis 18 weisen die Maschine an, ein Rechteck von 2 cm Kantenlänge mit minimaler Ausgangsleistung zu fahren. Der Befehl 'G0' in Zeile 19 weißt die Maschine an, sich zügig an die Startposition zu begeben.
Auf diese Art und Weise kann man die Maschine ohne ein weiteres Hilfsprogramm präzise steuern. Die Voraussetzung für die Präzision ist natürlich, dass die Maschine richtig kalibriert ist.

3.2.2 Kalibrieren des LC

Die Kalibrierung eines LC mittels der Software LightBurn ist sehr einfach und sollte die einzige Möglichkeit sein, wie ein Lasercutter einzustellen ist.
Man definiert ein Quadrat von vorzugsweise 100 x 100 mm Kantenlänge, und lässt dieses aus einem festen, aber nicht harten Material ausschneiden. Für das Beispiel wurde 4 mm Pressspan Fichtenholz genommen. Nachdem das Werkstück sauber ausgeschnitten wurde, misst man die Kantenlänge mittels Messschieber nach. Im ersten Schnitt waren es auf der x-Achse 99,45 mm und der y-Achse 99,75 mm.

Im zweiten Schritt nutzt man LightBurn und geht unter Einstellungen → Maschineneinstellungen → Klickt auf 'Einlesen', um die eingestellten Daten aus den LC einzulesen und dann auf 'Achsen kalibrieren'. Hier gibt man in die entsprechenden Felder den Sollwert von 100 mm und den Istwert von 99,45 bzw 99,75 mm ein. LightBurn korigiert die Einstellungen autonom und man erhält beim nächsten Schnitt ein präzise ausgeschnittendes Werkstück.

Abbildung 11: Werksstücke für Kalibrierung

Hierbei sei erwähnt, dass ich beim ersten Versuch wohl einen Fehler gemacht habe und eine zweite Kalibrierung vornehmen musste. Diese misst aber genau 100,00 mm auf beiden Achsen mit einem absoluten Messfehler des Messschiebers von $\Delta l = 0,05mm$. Somit liegt der Fehler außerhalb des durch mich messbaren Bereiches und muss kleiner als 0,05 mm sein. Da für diese Arbeit eine absolute Abweichung von dem Zehnfachen der Messgenauigkeit angesetzt ist, wird dieser Wert als hinreichend genau angesehen.

3.2.3 Den LC richtig einstellen

Es gibt drei Variablen, die man einstellen muss, damit ein LC einen sauberen Schnitt erstellen kann. Diese drei Variablen sind:

- Die Schneidegeschwindigkeit, mit der der LC dich schneidend zurch das Maretial bewegt. Angebeben wird diese Variabel in mm / min.
- Die Ausgangsleistung in % Abhängig von der Maximalen Leistung, die das Gerät zur Verfügung stellen kann.
- Die Wiederholrate des Schneidvorganges ohne Einheiten.

Trotz einer intensiven Studie von Lehrbüchern und Internetquellen bleibt es viel mehr ein 'Erfühlen' der richtigen Kombination der drei Parameter, als ein Wissen.
Falls man die drei Variablen nicht auf das zu schneidende Material anpasst, gibt es zusammengefasst zwei mögliche Ergebnisse:

- Das Material ist angeschnitten, aber nicht durchgeschnitten.
- Das Material ist durchgeschnitten und zusätzlich an den Schnittkanten verkohlt.

Die "Extremwerte" für viel zu wenig Leistung pro Zeit auf die Schneidfläche ist, dass das Material unberührt ist. Bei zu viel Leistung pro Fläche auf Zeit ist nicht nur die Schnittkante, sondern das gesamte Material verbrannt.
LightBurn bietet hier die Möglichkeit eine n x m Matrix zu erstellen, mit der man zwei der drei Variablen in einem frei definierten Bereich verändern kann. Mit mindestens zwei dieser Matrizen kommt man auf eine gute Einstellung der Werte. Diese Methode ist sehr zeit- und materialintensiv, bedarf aber nur wenig Aufmerksamkeit durch die anwendende Person.
Die für diese Arbeit verwendete Arbeitsweise war hingegen eine andere. Die Leistung wurde permanent auf 95% gestellt, außer es waren sehr leicht entflammbare Materialien wie Papier dabei. In dem Fall boten sich 25% an. Die volle Leistung der Maschine wurde nicht ausgenutzt, da es sich um eine Laserdiode handelt und diese bei bei den letzten 5% kaum noch mehr Leistung abgeben kann, und stattdessen die aufgenommene Leistung in Wärme umwandelt. Es wurde zunächst ein längliches Werkstück aus dem zu schneidenden Material abgetrennt und in einem einzelnen Durchlauf die Schnittgeschwindigkeit so lange reduziert, bis ein tiefer Schnitt zu sehen war, aber das Material noch nicht gebrannt hat. Im zweiten Schritt wurde die Wiederholrate auf einen sehr hohen Wert gestellt. Als Beispiel nehme man 40. Das Werkstück wurde mit Unterlegscheiben etwas von dem Untergrund angehoben und ein dünner Streifen

vom Werkstück abgeschnitten, der nicht von den Unterlegscheiben getragen wurde. Der Schnitt war abgeschlossen, wenn der dünne Streifen abfiel. Der Schneidvorgang wurde beendet, die letzte Wiederholrate wurde notiert und der Versuch wiederholt, um ein sicheres Ergebnis zu erhalten. Diese Arbeitsweise verbraucht wenig Material und Zeit, bedarf aber ein höheres Maß an Aufmerksamkeit von der betreuenden Person.

Abbildung 12: Einstellen des LC

Theoretisch kann dieser LC einen Schnitt mit mehr als 700 mm / min durchführen. Dass dies nicht nur für die Mechanik, sondern auch für die Schnittqualität ein schlecht eingestellter Wert ist, zeigt folgendes Bild:

Abbildung 13: Vergrößerung eines Schnittmusters

Wenn der LC schnell eine harte Kurve fahren muss, beginnt er zu vibrieren und es braucht einige Zeit, bis die Vibration wieder nachlässt. Dies ist die 10fache Vergrößerung eines Schnittes mit 700 mm / min und einer Leistung von 100 % auf einem Amazon Karton. Die rechte Gravur ist 4 mm lang und zeigt eine deutliche Schwingung in Y Richtung auf.

3.3 Der Rahmen

Wenn ich in dieser Bachelorarbeit von einem Rahmen spreche, dann bezeichne ich damit das Werkstück auf dem das auszuschneidende Motiv fixiert wird. Der Rahmen besteht aus einer 40 cm x 26 cm großen Siebdruckplatte. Er wird mit der glatten Seite nach unten auf die Arbeitsfläche geschraubt. Die nun nach oben zeigende raue Oberfläche reflektiert das Licht weniger stark, was unseren Aufnahmen zugute kommt.

Während der Arbeit mit den LC verlässt die Siebdruckplatte ihre Position nicht. Auf die Siebdruckplatte wird mittels eines transparenten Klebestreifen ein DIN A4 großes, gebleichtes Blatt Papier geklebt. Dieses Blatt Papier sollte eine Festigkeit von etwa 120 g / m^2 haben. Der LC schneidet auf dem Blatt Papier 8 Kreise, die uns später als Referenzpunkte dienen sollen. Die Kombination aus dem weißen Papier und der dunklen Siebdruckplatte ermöglicht es, Bilder mit einem hohen Kontrast aufzunehmen. Die sich aus den ausgeschnittenen Kreisen, dem weißen Papier und der dunklen Siebdruckplatte, ergebende Einheit wird hier als Rahmen bezeichnet. Auf diesem Rahmen wird, mittels eines transparenten Klebestreifens, das auszuschneidende Motiv fixiert.
Die Konstruktion dieses Rahmens bietet einige Vorteile:

- Dieser Rahmen ist preiswert in der Anschaffung und schnell in der Fertigstellung. Dies macht Fehler in der Anwendung wenig dramatisch.
- Die Detektion von Kreisen ist technisch nicht besonders anspruchsvoll. Das ermöglicht einen einfachen Programmiercode.
- Die räumliche Ausdehnung in der Höhe ist bei diesem Rahmen zu vernachlässigen. Da der Laser eine Festbrennweite hat, und nur auf dieser Brennweite optimal arbeiten kann, kann der Laser vor Beginn der Arbeit auf den Abstand des Rahmens geeicht werden und muss danach nicht noch mal angepasst werden.

3.4 Ausrichtung des Aufgenommenen Bildes

Über MatLab gebe ich dem LC das Kommando, die acht Kreise für die Detektion des Rahmens auszuschneiden. Wie ich das genau bewerkstellige, möchte ich im Kapitel xx genauer beschreiben. Diese Vorgehensweise versetzt mich in die Position, dass ich genaue Informationen darüber habe, wo sich die acht Kreise im Koordinatensystem des LCs befinden. In diesem Kapitel geht es darum, die Brücke zwischen dem Koordinatensystem des LCs und der Bildmatrix zu schlagen. Auch mit Hilfe eines Stativs und einer fest installierten Kamera ist es besonders aufwendig, diese 100%tig plan zu der Ebene zu montieren, auf der sich das Papier befindet. Daher nehme ich an, dass das Bild freihändig aufgenommen wird und richte es über MatLab plan zur Papierebene aus. Zuerst beschreibe ich die Detektion der Kreise und die Ausrichtung des Bildes. Um die Präzision nachmessen zu können, benutze ich im ersten Schritt ein computererstelltes

Bildmaterial mit perfekt angeordneten Kreisen, welche ich mit MatLab um einen definierten Wert in X und Y drehe. Im zweiten Schritt präsentiere ich die angewandten Funktionen an einem realen Bild und zeige zugleich die verwendeten Filter, die ich benutze um das Bild auszuwerten.

3.4.1 Definition des Schnittmusters für den Rahmen

Das Schnittmuster für den Rahmen musste drei Bedingungen erfüllen:

- Es musste ein Muster beinhalten, das leicht von einem Programm erkannt werden kann
- Das Muster sollte präzise zu erkennen sein
- Das Muster sollte schnell mit dem LC angefertigt werden können.

Diesen drei Kriterien folgend optimierte ich meine Ideen immer weiter, bis ich mich schließlich für das Muster auf Abbildung 'Skizze vom Schnittmuster' entschieden habe.

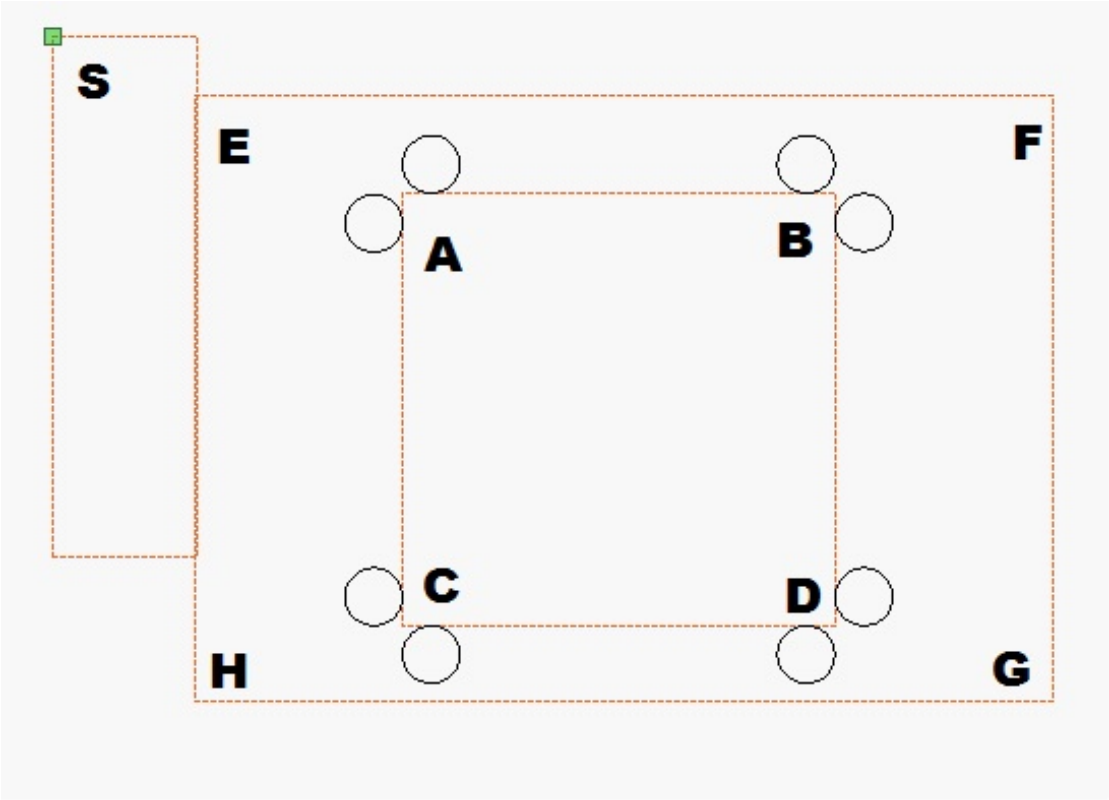

Abbildung 14: Skizze vom Schnittmuster

Der LC verfügt über keine Decoder so dass er nur Bewegungen fahren kann, die sich auf seine letzte Position beziehen. Somit definiere ich die Startposition meines LCs auf die obere linke Ecke. In der Skizze repräsentiert dieser Punkt das grüne Quadrat mit der Bezeichnung S. Das Rechteck , in dem sich die Markierung S befindet, dient nur der räumlichen Differenzierung zwischen dem Startpunkt und dem DIN A4 Blatt. Das DIN A4 Blatt, welches mit einem durchsichtigen Klebestreifen an die umgedrehte Siebdruckplatte befestigt ist, wird hier von dem Rechteck 'EFHG' beschrieben. Das Quadrat 'ABCD' ist zentriert auf das Quadrat 'EFGH' gelegt. An dem Eckpunkt 'ABCD' befinden sich jeweils zwei Kreise mit einem Durchmesser von 20 mm. Der Mittelpunkt der Linie von einem Mittelpunkt des Kreises zum anderen liegt genau auf einen Eckpunkt des Quadrates 'ABCD'. Dieses Quadrat hat eine Seitenlänge von 150 mm. Dieses Quadrat beschreibt die Fläche, auf die später das Motiv zum Ausschneiden gelegt werden kann. Die schwarz umrundeten Kreise werden vom LC ausgeschnitten. Die orangenen Linien, sowie die Punktbenennungen spielen in dem Schneideprozess keine Rolle.

Durch das Finden der ausgeschnittenen Punkte auf der Bildmatrix und der Bestimmung der Punkte 'ABCD' kann ich die Bildmatrix nach den Strecken des Quadrates ausrichten und somit die Unebenheiten der Aufnahme korrigieren. Das Ergebnis dieser Prozedur soll eine Bildmatrix sein, die plan zur Oberfläche des Papiers ist. Um dieses Ergebnis simulieren zu können, erstellte ich die Bilddatei mit den Namen Á14'

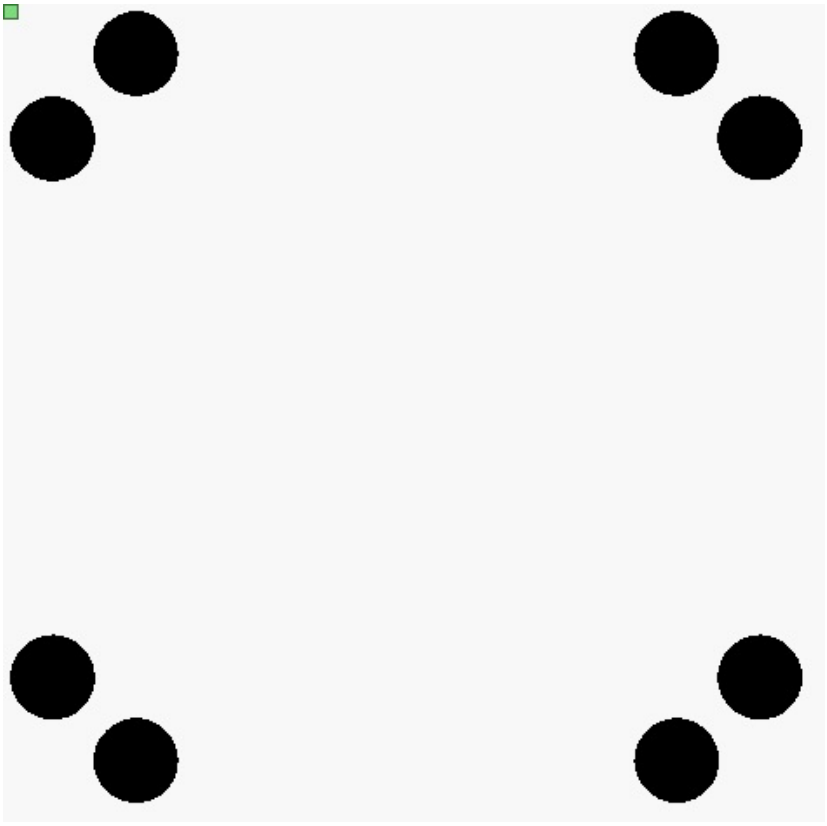

Abbildung 15: A14

3.4.2 Bildliche Erkennung der Ausgeschnittenen Kreise

In diesen Abschnitt beschreibe ich, wie ich die schwarzen Kreise auf einem Bild erkenne, den kleinsten Abstand zwischen den Kreisen bestimme und auf der Linie die Punkte 'ABCD' herausfinde.

Der Code 'Einlesen und Verschieben von einem Bild' zeigt, wie ich das Bild 'A14' in den Code integriere und ihn dann mittels zufällig gewählter Variablen in X und y-Richtung verschiebe. Diese Verschiebung wie g simuliert schief halten des Smartphones über der Papierebene mit dem großen Vorteil, dass die Verschiebung mit den Variablen rx und ry genau definiert ist und später mit den Korrekturwert verglichen werden kann.

```
%% bild Verschieben
anfang=imread('A14.jpg');
a = 0; b = 0.05;
rx = a + (b-a).*rand;
ry = a + (b-a).*rand;
tform = maketform('affine',[1 rx 0; ...
```

```
                            ry 1 0; ...
                            0 0 1]);
anfang = imtransform(anfang,tform);
bild=rgb2gray(anfang);% Grau-stufe
clear figure
imshow(bild)
hold on
```

Quellcode 2: Einlesen und Verschieben von einem Bild

Das eingelesene und um

rx = 0.047875341771715 und

ry = 0.048244426759964

verschobene Bild A14 zeigt die folgende Darstellung 'A14 verschoben'

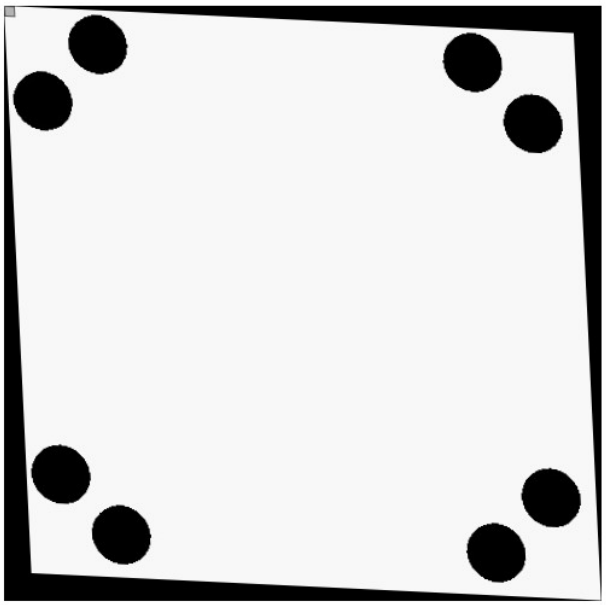

Abbildung 16: A14 Verschoben

Mit dem nun folgenden Code [9]finde ich die Kreise in der Bild Matrix und gebe ihren Mittelpunkt grafisch aus.

```
%% Fläche, Zentrum, Durchmesser, Umfang der Objekte auslesen:

daten=regionprops(bild,'Area','Centroid',...
            'EquivDiameter','Perimeter');

%% Nur Kreise nehmen, die rund genug sind.

rund_obj=daten(4*pi.*[daten.Area]./[daten.Perimeter].^2 >=0.73);
clear figure
imshow(bild);
hold on;


%% finde den Mittelpunkt von allen Kreisen
clear figure
imshow(anfang)
for i=1:length(rund_obj)
    m2 = rund_obj(i).Centroid;
    hold on
    plot(m2(1), m2(2),'r*')
end
clear i m2
```

Quellcode 3: Erkennen von Kreisen in der Bildmatrix und den Mittelpunkt Grafisch Darstellen

Auf der Grafik 'A14 mit Kreismittelpunkten' ist zu erkennen, dass die Software die Kreise gut erkannt hat und deren Mittelpunkte deutlich bestimmt hat.

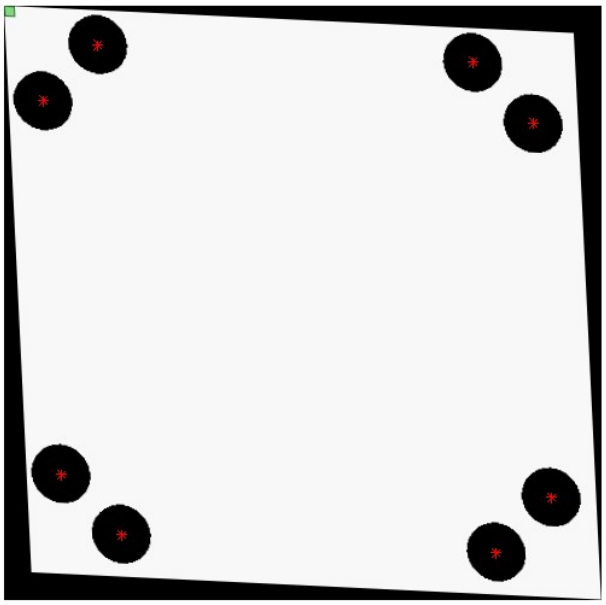

Abbildung 17: A14 mit Kreismittelpunkten

Nun gilt es herauszufinden, welcher Mittelpunkt den kleinsten Abstand zu einem anderen Mittelpunkt hat. Diese beiden Mittelpunkte sind als ein Paar zu betrachten und sollen für die weitere Suche keine Rolle mehr spielen. Der nun folgende Code [10] löst dieses Problem und das Ergebnis ist in Abbildung ' A14 mit Kreispaaren' zu erkennen.

```
%% kleinsten Abstan Finden
clear Dist Index l figure
KreisPaarGeringerAbstandX = zeros(2);
KreisPaarGeringerAbstandY = zeros(2);
o = 1;

for k = 1:size(rund_obj)
    l(k) = k;
end
imshow(anfang)
hold on
for j = min(l):size(rund_obj)-1
    j= min(l)
    clear Dist Index
    for k = l(2):size(rund_obj)
        k
        A = rund_obj(j).Centroid;
        B = rund_obj(k).Centroid;
        x1 = A(1);   y1 = A(2);
        x2 = B(1);   y2 = B(2);
        Dist(k) = sqrt((x2 - x1)^2 + ( y2 - y1 )^2)
        x = [A(1) B(1)]; y = [A(2) B(2)];
        plot(x,y,'Color','blue','LineStyle','--')
    end
    Dist(Dist==0)=[] %https://de.mathworks.com/matlabcentral/answers/176196-how-to-
        delete-a-zeros-in-matrix
    [M,I] = min(Dist) \
    j
    l(I+1)
    A = rund_obj(j).Centroid;
    plot(A(1),A(2),'r*')
    B = rund_obj(l(I+1)).Centroid; %es ist der Zweite Stich aber die Dritte Kugel
        die die kürzeste Verbindung hat
    plot(B(1),B(2),'r*')
    x1 = A(1);   y1 = A(2);
    x2 = B(1);   y2 = B(2);
    x = [A(1) B(1)]; y = [A(2) B(2)];
    plot(x,y,'Color','red','LineStyle','--')
```

```
37      KreisPaarGeringerAbstandX(o,:) = x;
38      KreisPaarGeringerAbstandY(o,:) = y;
39      o = o + 1;
40
41      l(l==(l(I+1)))=[]; % next i+1 versuchen
42      l(l==j)=[];
43      if isempty(l)
44          break;
45      end
46  end
47
48  clear A ans B Dist I j k l M o x x1 x2 y y1 y2
```

Quellcode 4: Finden der kleinsten Abstände zwischen den Mittelpunkten der Kreise

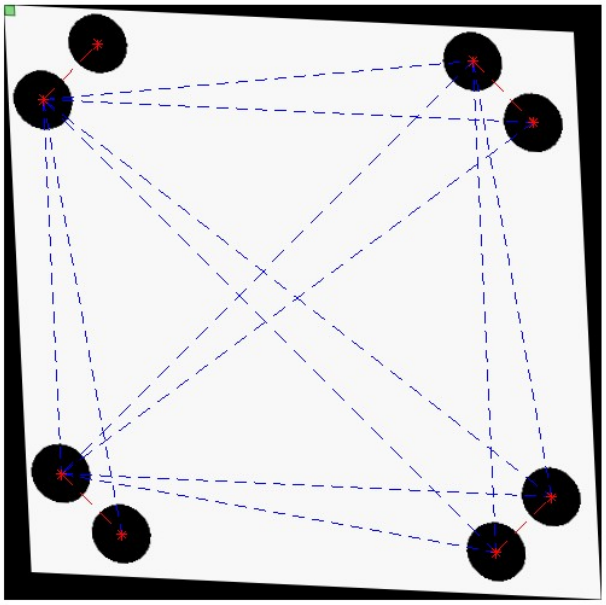

Abbildung 18: A14 kleinster Abstand zwischen den Mittelpunkten finden

Nachdem die zusammengehörigen Punkte als Paare definiert wurden, ist es nun ein Leichtes auf der Hälfte der Strecke zwischen den beiden Mittelpunkten von den Kreisen einen weiteren Mittelpunkt von der Strecke zu setzen. Der nun folgende Code setzt dies um und das Bild 'A14 mit Punkten ABC' zeigt das Ergebnis.

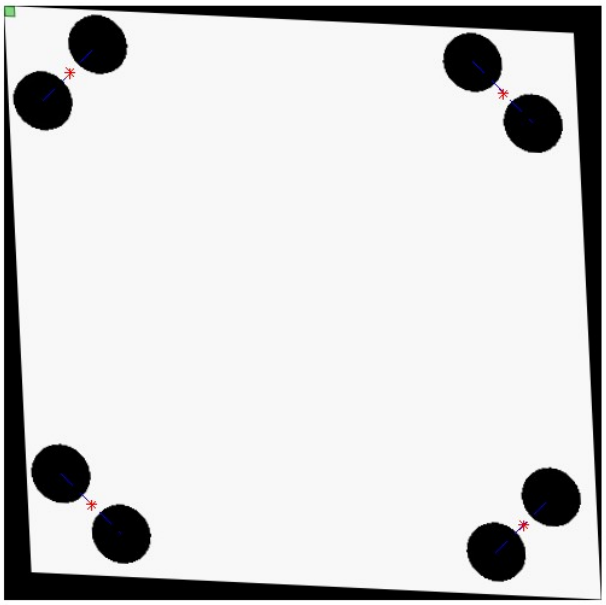

Abbildung 19: A14 mit den Punkten ABCD

3.4.3 Bildmatrix vertikal und horizontal ausrichten

Die Punkte 'ABCD' sind nun definiert und im folgenden Abschnitt wird das Bild vertikal und horizontal ausgerichtet. Um dieses Ergebnis zu erreichen, definiere ich zwei diagonal zueinander stehende Punkte als Fixpunkte und die beiden anderen Punkte werden vertikal und horizontal an die Position der beiden Fixpunkte angepasst. Berechnet werden die einzelnen Strecken AB , BC, CD und DA sowie der horizontale und vertikale Abstand zwischen den Soll-Punkten und den zu verschiebenden Punkten. Die Punkte A und C lassen sich besonders einfach finden, indem ich auf die Matrix mit den Eckpunkt eine Minimum und Maximum Funktion anwende. Daher habe ich mich entschieden, diese beiden Punkte als Fixwerte anzunehmen.

```
% A und C Definieren
[MaxWert MaxIndex] = max(EckPunkte)
[MinWert MinIndex] = min(EckPunkte)
```

Quellcode 5: Finden Der Punkte A und C

Im Anschluss habe ich aus der Matrix EckPunktCop die Min und Max Werte heraus gelöscht und so hatte ich einen einfachen Zugang zu den Fixwerten und den Punkten

die verändert werden müssen.[2]

Nun ist alles zusammen, um die Winkel zu berechnen, mit deren Hilfe das Bild wieder horizontal und vertikal ausgerichtet werden kann. Im Quellcode 'Strecken berechnen und Sollpunkte definieren' wurden die Sollpunkte mit (oben rechts) ORSollPunkt und (unten links) ULSollPunkt berechnet. der Abstand von den Punkten B und C von den Sollpunkten wurde mit den Berechnungen XverschiebungUntenLinks bis YverschiebungObenRechts berechnet. Mit denen sich hieraus gebildeten Strecken werden zwei rechtwinklige Dreiecke parallel zur x- und y-Achse mit den Punkten B und C gebildet. Der Winkel[3], mit dem die Strecke AB von der y-Achse abweicht, ist unser gesuchter Korrekturwert in x-Richtung. Die Abweichung und y-Richtung errechnet sich durch den Winkel der Strecke CB in Abhängigkeit zur y-Achse. Die Abbildung - ABCD als Viereck mit Sollwerten - zeigt noch mal deutlich die Verschiebung der Bildmatrix und die nötige Korrektur.

[2]https://www.gomatlab.de/betrag-vom-vektor-t31234.html
[3]https://de.mathworks.com/help/matlab/ref/atan.html

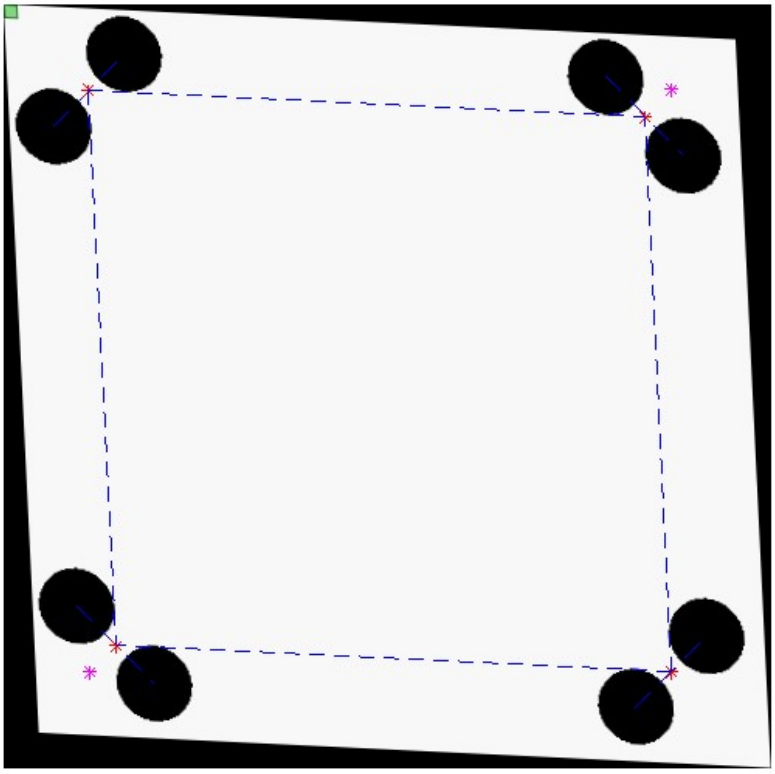

Abbildung 20: ABCD als Viereck mit Sollwerten

```
% Rotation des Bildes

f = anfang;
clear figure
imshow(f)

c = YLinks
b = YVerschiebungUntenLinks
ryRet = sin(b/c)
c = XOben
b = XVerschiebungObenRechts
rxRet = sin(b/c)

T = [1 -rxRet 0;
     -ryRet 1 0;
     0 0 1 ];
```

```
17 tform = maketform('affine',T);
18 g1 = imtransform(f,tform);
19 imshow(g1);
20
21 clear f s T tform theta
```

Quellcode 6: Korigirendes Routieren der Bildmatrix

Die berechneten Werte für die Rückrotation sind:

rxRet = 0.047765943792164 und

ryRet = 0.048182655891133

Das Ergebmis der Rotation zeigt Abbildung: - Bildmatrix Ausgerichtet -.

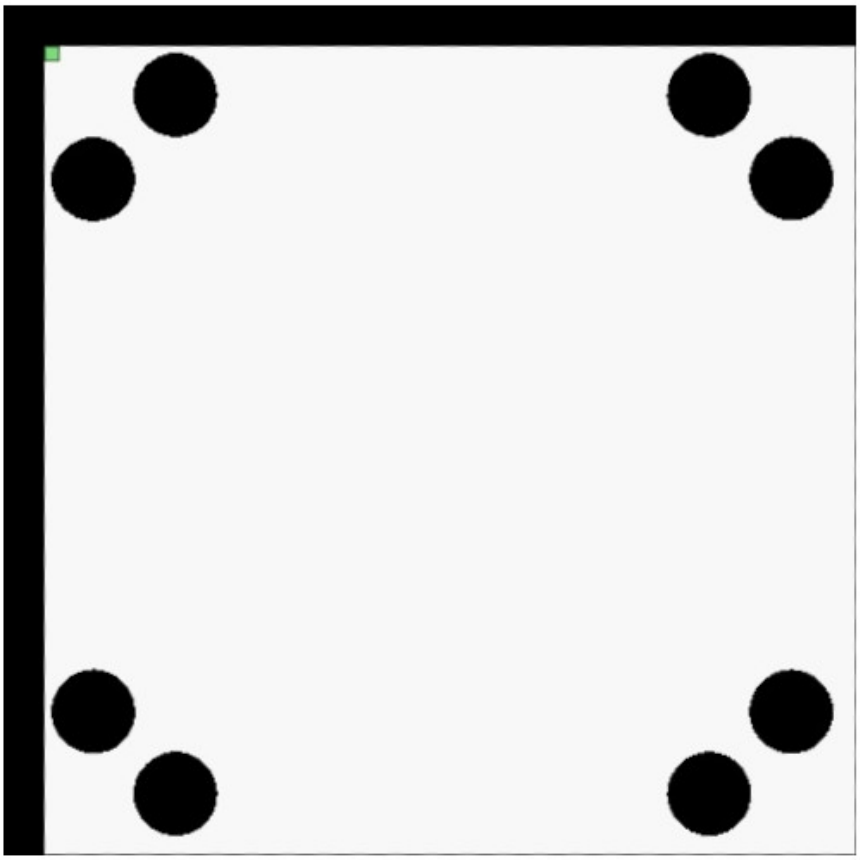

Abbildung 21: Bildmatrix Ausgerichtet

Ein Vergleich von den Werten rx und ry mit rxRet und ryRet zeigt folgendes[4]:

	x	y
Vorgabe in rad	0.04787	0.04824
Berechnung in rad	0.04776	0.04818
Differenz in rad	0.00010	0.00006
relative Differenz in %	0.22850	0.12803

3.4.4 Bestimmung der Genauigkeit

Im letzten Absatz habe ich gezeigt, dass mein Messverfahren eine relative Abweichung von einem Prozent aufweist. In diesen Absatz möchte ich belegen, dass dies eine hinreichende Genauigkeit ist, um dieses Projekt weiterzuführen. Die Seitenlänge, auf die der LC präzise arbeiten soll, ist mit A = 150 mm definiert.
Die Absolute Abweichung, die der Schnitt des Systems haben soll, beträgt A = 0,5 mm.
Die absolute Abweichung, die der LC von Herstellerseite aus mitbringt, liegt bei: 0,01 mm.

Wenn eine Schnittlänge die maximal mögliche Länge von S = 150 mm aufweist, wird sie vom System mit einer absoluten Ungenauigkeit von±S = 0,33 mm angegeben. Addiert man noch die Ungenauigkeit vom LC hinzu, kommen wir auf eine maximale Abweichung von S = 0,34 mm.
Dies ist nur eine Beispielrechnung mit einem Wertepaar. Doch sollte das Ergebnis ausreichen, um behaupten zu können, dass das Messsystem hinreichend genau ist, um die geforderten Aufgaben bewältigen zu können.

[4]https://glossar.item24.com/glossarindex/artikel/item/messabweichung.html: :text=Die%20relative%20Abweichung%20ist%

3.5 Kantenerkennung im Motiv

An dieser Stelle wird über die Kantenerkennung in einem aufgenommenen Bild geschrieben. Durch gründliche Auswahl der ausführenden Software, MatLab, ist dieses Unterkapitel leicht zu behandeln.

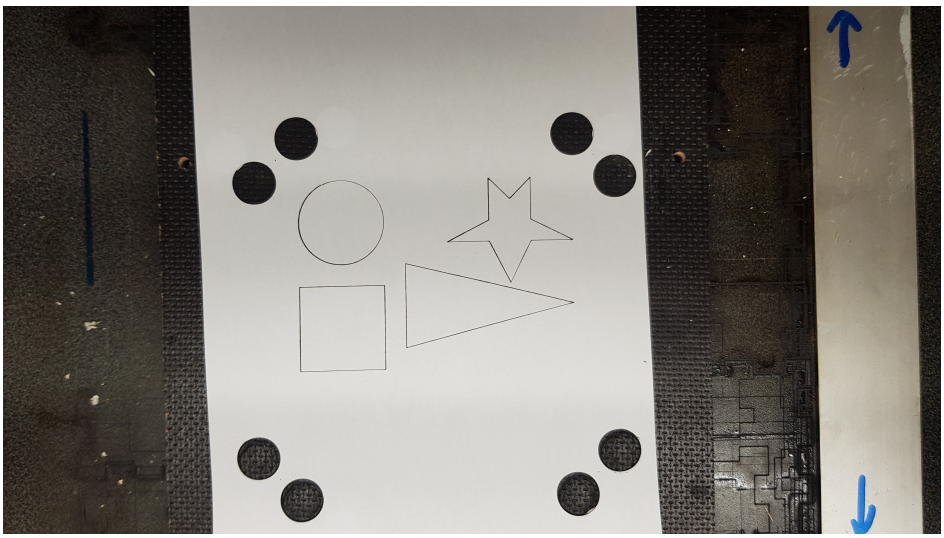

Abbildung 22: Einfaches Kontrastreiches Motiv

Die Aufnahme 'Einfaches kontrastreiches Motiv' zeigt einfache geometrische Figuren, die ich mit dem Laser Cutter in das Papier gebrannt habe. Dadurch ist eine feine schwarze Linie entstanden, die auf dem Bild gut zu erkennen ist und klare Kanten aufweist.

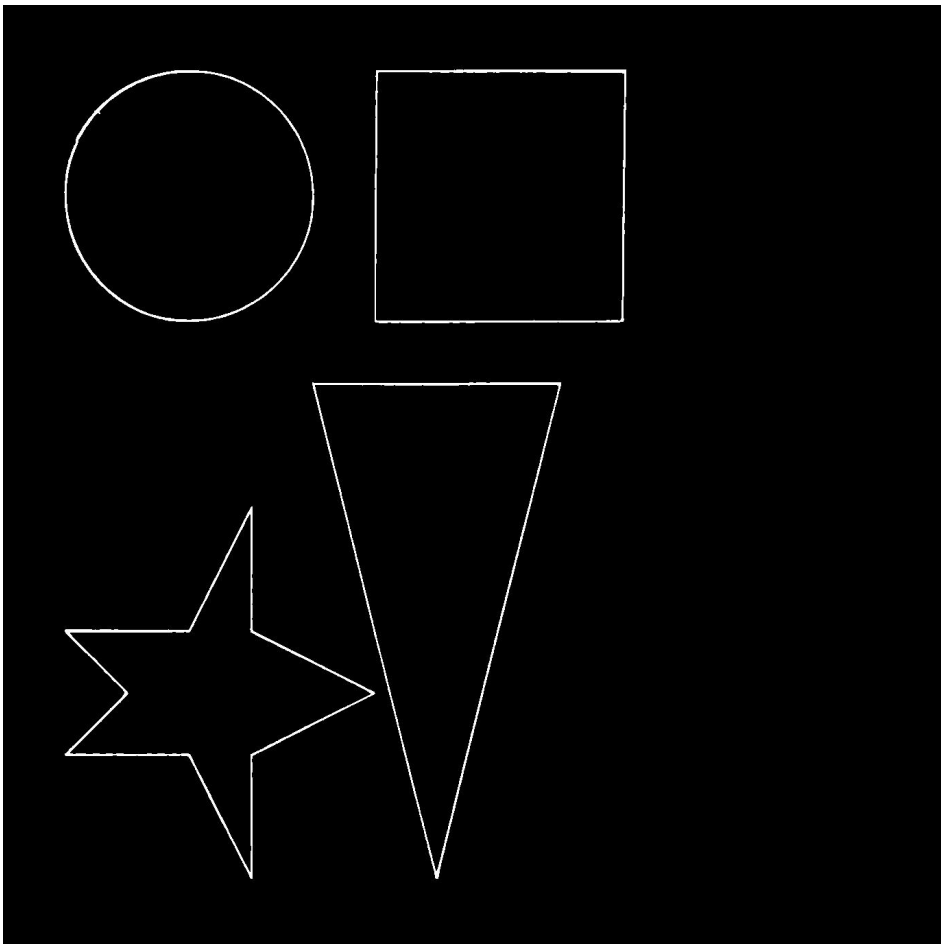

Abbildung 23: Motiv nach Kantenerkennung

MatLab bietet eine Vielzahl von Kantenerkennungsalgorithmen an, die je nach Größe der Oberfläche und Struktur des Motivs individuell ausgewählt werden können. In diesem Fall, bei einem trivialen Motiv und maximalem Kontrast, ist die Wahl des Algorithmus nahezu beliebig, da die Kanten leicht erkennbar sind.

3.6 Generierung von CNC-G-Code aus binären Bildern

Nach der Erzeugung einer eindeutigen Schnittkante folgt nun die Generierung des G-Codes für den LC. Die Funktion `GenerateGCode` erhält ein Bild als Eingangsvariable und generiert eine neue Datei mit dem Namen `LCBildAusschneiden_heutigesDatum_FortlaufenerIndex.g`, in die der passende Bewegungsablaufplan für das Bild geschrieben wird. Der genaue Abstand wird in der Funktion aus der Bildgröße berechnet. Diese wurde für die Funktion auf das Rechteck ABCD beschränkt. So kann der Funktionsaufruf `pixelSize = calculatePixelSize(image)`

aus der definierten Seitenlänge des Quadrats und der Größe und Breite des Bildes in Pixeln den Maßstab berechnen, in dem der G-Code in der `while`-Schleife dimensioniert werden soll.

```
function generateGCode(image)

    % Berechnen des Abstands zwischen den Punkten in Pixeln
    pixelSize = calculatePixelSize(image);
  %  pixelSize = 1;
    %  ffnen  der G-Code-Datei zum Schreiben
    % Generiere Dateinamen
    filename = generateFilename('LCBildAusschneiden','g');
    fileID = fopen(filename, 'w');

    % Schreiben der G-Code-Anweisungen in die Datei
    fprintf(fileID, 'G00 G17 G40 G21 G54\n'); % Einstellen des metrischen
        Koordinatensystems
    fprintf(fileID, 'G90\n'); % Absolute Positionierung
    fprintf(fileID, 'M8\n'); % Einschalten der Spindel

    % Anfahren des Startpunkts
    StartX = 100;
    StartY = -100;
    fprintf(fileID, 'G0 X%.2f Y%.2f\n', StartX, StartY);
    fprintf(fileID, 'M3\n'); % Absolute Positionierung

    % Finde den ersten Pixel
    [row, col, command] = findNearestPoint(image, [0 , 0]);
    % Abfrage nach der intensität
    zahl = input('Mit welcher Leistung soll das Bild geschnitten werden: (in %)');
    zahl = zahl * 10;
    fprintf(fileID, 'S%.2f F550\n',zahl);
    % Aktiviere den Parallelpool
    parpool();
    while white_points_exist(image) && ~isempty(command) % berprüfe, ob weiße Punkte
         im Bild existieren und ein Befehl vorhanden ist
        fprintf(fileID, '%s X%.2f Y%.2f\n', command, StartX + row * pixelSize,
            StartY + col * -pixelSize);

```

```
        % Aktualisiere das Bild, indem der gefahrene Punkt entfernt wird
        image(row, col) = 0;

        % Finde den nächsten Pixel
        [row, col, command] = findNearestPoint(image, [col, row]); %Orginal
       % [row, col, command] = findNearestPoint(image, [row, col]); %änderung
           20230624

    end

    % Zurück zum Startpunkt
    fprintf(fileID, 'G0 X00.00 Y00.00 S0F1000');

    % Schließen der Datei
    fclose(fileID);
    % Schließe den Parallelpool
    delete(gcp);

    disp(['G-Code wurde erfolgreich generiert und in der Datei ' filename '
        gespeichert.']);
end
```

Der Rahmen wurde so manipuliert, dass der Punkt A auf der Position X100 Y-100 liegt, sodass der Nullpunkt des übergebenen Bildes hier einfach als Startpunkt übergeben werden kann.
Nach dem Schreiben einiger einstellender Kommandos und der Abfrage, mit welcher Leistung der LC arbeiten soll, springt die Funktion in die primäre While-Schleife, in der immer der nächste weiße Pixel im Bild gesucht wird und seine Position als Kommando in die G-Datei eingetragen wird.

3.7 Simulation der Ausgabe

Nach den ersten drei generierten G-Code-Bildern wurde mir schnell klar, dass man wesentlich schneller und ressourcenschonender ans Ziel kommt, wenn man sich eine Simulationsumgebung aufbaut.

Meine CNC-Simulation musste folgende Bedingungen erfüllen: Sie musste in der Lage

sein, zwei G-Code-Dateien einzulesen und in unterschiedlichen Farben darzustellen. Außerdem musste sie in der Lage sein, G-Code zu lesen, der mit relativen und absoluten Distanzen arbeitet. Der folgende Code erfüllt diese Bedingungen:

```
function CNC_Maschinen_Simulationen(gcode_file1, gcode_file2)
    % Aktiviere den Parallelpool
    parpool();

    % Lese die G-Code-Dateien ein
    gcode_lines1 = importGCode(gcode_file1);
    gcode_lines2 = importGCode(gcode_file2);

    % Initialisiere die Position auf dem Arbeitsbereich
    currentPosition = [0, 0];

    % Zeichne das Bild
    figure;
    hold on;

    % Setze den Betriebsmodus auf G90 (Absoluter Modus) zu Beginn
    absoluteMode = true;

    % Zeichne das erste G-Code-File (Magenta-Rot)
    for i = 1:numel(gcode_lines1)
        line = gcode_lines1{i};
        disp(['␣' line '']);
        % disp(line)

        % Extrahiere den Befehl und die Koordinaten aus der G-Code-Zeile
        [command, xCoordinate, yCoordinate] = parseGCodeLine(line);
        coordinates = [xCoordinate, yCoordinate];

        % berprüfe den Befehl und aktualisiere den Betriebsmodus
        if strcmp(command, 'G90')
            absoluteMode = true; % Setze auf absoluten Modus (G90)
            continue; % Springe zum nächsten G-Code-Befehl
        elseif strcmp(command, 'G91')
            absoluteMode = false; % Setze auf inkrementalen Modus (G91)
            continue; % Springe zum nächsten G-Code-Befehl
```

```
        end

        % Führe die entsprechende Berechnung basierend auf dem Betriebsmodus aus
        if absoluteMode
            % Im absoluten Modus (G90) werden die Koordinaten als absolute Positionen
                interpretiert
            newPosition = coordinates;
        else
            % Im inkrementalen Modus (G91) werden die Koordinaten als Verschiebung
                von der aktuellen Position interpretiert
            newPosition = currentPosition + coordinates;
        end

        % Berechne die Wegstrecke vom aktuellen Punkt zum Zielpunkt
        distance = norm(newPosition - currentPosition);

        % Zeichne den Laserstrahl in Magenta-Rot
        plot([currentPosition(1), newPosition(1)], [currentPosition(2), newPosition
            (2)], 'm');

        % Aktualisiere die aktuelle Position
        currentPosition = newPosition;

        % Simuliere die Lasercutter-Brennzeit basierend auf der Wegstrecke
        burnTime = distance / 1000; % Beispielhafte Annahme: 1 mm/s
            Brenngeschwindigkeit

        % Warte entsprechend der Brennzeit
        % pause(burnTime);
    end

    % Zeichne das zweite G-Code-File (Blau-Grün)
    for i = 1:numel(gcode_lines2)
        line = gcode_lines2{i};
        disp(['␣' line '']);
        % disp(line)

        % Extrahiere den Befehl und die Koordinaten aus der G-Code-Zeile
        [command, xCoordinate, yCoordinate] = parseGCodeLine(line);
```

```
71        coordinates = [xCoordinate, yCoordinate];
72
73        % berprüfe den Befehl und aktualisiere den Betriebsmodus
74        if strcmp(command, 'G90')
75            absoluteMode = true; % Setze auf absoluten Modus (G90)
76            continue; % Springe zum nächsten G-Code-Befehl
77        elseif strcmp(command, 'G91')
78            absoluteMode = false; % Setze auf inkrementalen Modus (G91)
79            continue; % Springe zum nächsten G-Code-Befehl
80        end
81
82        % Führe die entsprechende Berechnung basierend auf dem Betriebsmodus aus
83        if absoluteMode
84            % Im absoluten Modus (G90) werden die Koordinaten als absolute Positionen
                  interpretiert
85            newPosition = coordinates;
86        else
87            % Im inkrementalen Modus (G91) werden die Koordinaten als Verschiebung
                  von der aktuellen Position interpretiert
88            newPosition = currentPosition + coordinates;
89        end
90
91        % Berechne die Wegstrecke vom aktuellen Punkt zum Zielpunkt
92        distance = norm(newPosition - currentPosition);
93
94        % Zeichne den Laserstrahl in Blau-Grün
95        plot([currentPosition(1), newPosition(1)], [currentPosition(2), newPosition
              (2)], 'b');
96
97        % Aktualisiere die aktuelle Position
98        currentPosition = newPosition;
99
100       % Simuliere die Lasercutter-Brennzeit basierend auf der Wegstrecke
101       burnTime = distance / 1000; % Beispielhafte Annahme: 1 mm/s
              Brenngeschwindigkeit
102
103       % Warte entsprechend der Brennzeit
104       % pause(burnTime);
105   end
```

```
106
107     hold off;
108     axis equal;
109     title('CNC Maschinen Simulation');
110     xlabel('X-Achse');
111     ylabel('Y-Achse');
112     % Schließe den Parallelpool
113     delete(gcp);
114 end
115
116
117 function gcode_lines = importGCode(gcode)
118     % Lese den G-Code aus einer Datei ein
119     fid = fopen(gcode, 'r');
120     gcode_lines = textscan(fid, '%s', 'Delimiter', '\n');
121     fclose(fid);
122
123     gcode_lines = gcode_lines{1};
124 end
125
126 function [command, xCoordinate, yCoordinate] = parseGCodeLine(line)
127     % Extrahiere den Befehl aus einer G-Code-Zeile
128     command = '';
129     if startsWith(line, 'G')
130         command = strtok(line);
131     end
132
133     % Extrahiere die Koordinaten aus einer G-Code-Zeile
134     xCoordinate = 0;
135     yCoordinate = 0;
136
137     % Suche nach dem Buchstaben 'X' im line-String
138     xIndex = strfind(line, 'X');
139     if ~isempty(xIndex)
140         % Extrahiere den X-Koordinatenwert
141         xCoordinate = sscanf(line(xIndex+1:end), '%f', 1);
142     end
143
144     % Suche nach dem Buchstaben 'Y' im line-String
```

```
145     yIndex = strfind(line, 'Y');
146     if ~isempty(yIndex)
147         % Extrahiere den Y-Koordinatenwert
148         yCoordinate = sscanf(line(yIndex+1:end), '%f', 1);
149     end
150 
151     % berprüfe auf NaN-Koordinaten und gebe eine Warnung aus
152     if isnan(xCoordinate) || isnan(yCoordinate)
153         warning('Eine oder mehrere Koordinaten sind NaN.');
154     end
155 end
```

Mit diesen Werkzeugen war ich in der Lage, schnell bestehende logische Programmierfehler zu beseitigen und meinen Code zu vervollständigen.
Hier zeigt es sich erneut von Vorteil, dass ich ein Motiv gewählt habe, welches ich mit dem LC in das Papier brennen konnte. Das gewählte Motiv kann in der Simulation ebenso gut als Vorlage verwendet werden und dient als Vergleich zwischen dem Original und dem erstellten Bild.

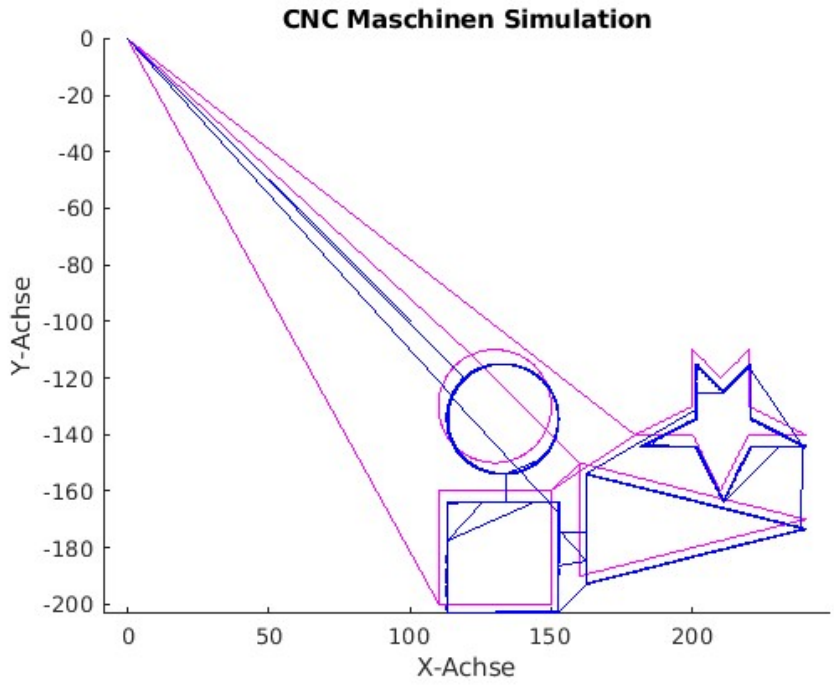

Abbildung 24: Original und generierter G-Code im Vergleich

Auf der Abbildung 'Original und generierter G-Code im Vergleich' ist zu erkennen,

dass schon viele Faktoren für ein sauberes Ausschneiden der Objekte gegeben sind: Die Objekte haben die gleiche Ausrichtung und Rotation. Zudem haben sie eine identische Größe und Form in X- und Y-Richtung. Letzteres lässt darauf schließen, dass die Perspektivkorrektur fehlerfrei funktioniert. Unschwer ist jedoch eine leichte Verschiebung in +X und -Y zu erkennen, die auch nach sehr sauberem Aufnehmen des Motives und sehr gründlicher Überprüfung der Berechnungen einfach nicht verschwand. Hierbei ist es übrigens gleichgültig, ob das Ergebnis von der Simulation oder vom LC erzeugt wird, beide Systeme zeigten das gleiche Verhalten.
Um die Fehlerursache gut darstellen zu können, skizziere ich hier noch einmal den Ablauf, der zu diesem Bild führt:

1. Ein Blatt Papier wird auf die Arbeitsfläche des LC gelegt und so fixiert, dass es nicht durch Berührung meinerseits oder die Belüftung des LC verschoben werden kann.

2. Der Rahmen wird auf das Papier gebrannt.

3. Die ausgeschnittenen Kreise werden entfernt.

4. Das 'Motiv' wird eingebrannt.

5. Es wird ein Foto vom Motiv und Rahmen gemacht und dieses an Matlab übergeben.

6. Über Matlab wird eine Perspektivkorrektur durchgeführt und das Bild 'gerade gerückt'.

7. Aus dem Bild wurde das Quadrat ABCD ausgeschnitten. Dies wurde in ein logisches Bild umgewandelt, Rauschen herausgefiltert und eine Kantenerkennung durchgeführt.

8. Aus dem Bild wurde ein G-Code generiert und dieser wurde mit dem Original verglichen.

Der letzte Fehler, der uns hier von der endgültigen Lösung abhält, ist, dass das Quadrat ABCD nach der Perspektivkorrektur neu gefunden werden muss, da sich die Position des Quadrates durch die Perspektivkorrektur verschoben hat. Die Erkennung der Kreise des Rahmens muss also für eine saubere Detektion zweimal erfolgen.

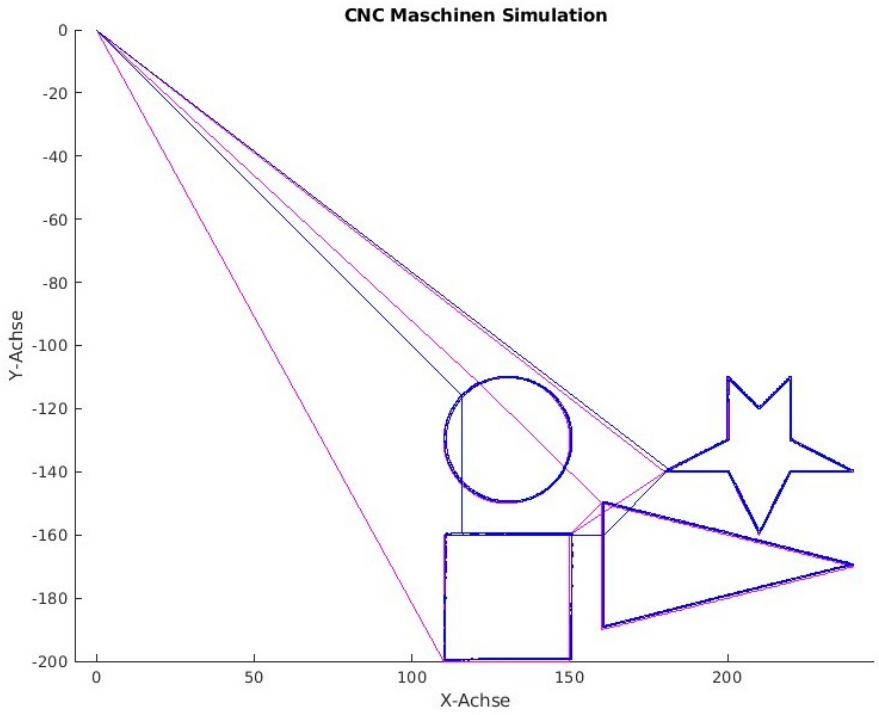

Abbildung 25: Original und generierter G-Code im finalen Vergleich

4 Ergebnisse

In dem nun folgenden Kapitel möchte ich die Ergebnisse dieser Arbeit darstellen und am Ende jedes Unterkapitels kritisch bewerten. Unterteilen will ich die Ergebnisse in Hard- und Software. Gegliedert sind die Unterkapitel in eine Gegenüberstellung dessen, was am Anfang des Projektes das Ziel war, und dem, was am Ende das Ergebnis ist.

4.1 Schneideergebnisse

Die Zielsetzung der Arbeit war es, ein Schneidegerät mit einer Genauigkeit von weniger als 0,5 mm mit einem Computer zu verbinden und diesen präzise ansteuern zu können. Vergleiche hierfür Kapitel: 2.2 'Anforderungen an das System und dessen Konfiguration' Aufzählungspunkt 3.
Mit dem 'S9 Sculpfun' in Kombination mit der einfachen Bedienbarkeit über Lightburn und der zusätzlichen Ansteuerung über Matlab ist es möglich, den LC einfach zu kalibrieren, sodass Punkt 4 in der Kriterienliste von Kapitel 2.2 ebenfalls erfüllt ist.

Das Verbinden von Computer und Lasercutter, sowie das präzise Ansteuern und Bedienen dieses Gerätes ist in dieser Arbeit gelungen.

4.2 Einlesen und Ausrichten des Bildes

Das Einlesen und das Ausrichten des Bildes sind zwei elementare Schritte, um das Ziel der Arbeit zu erreichen. Daher wurde ihnen auch ein Großteil der Zeit und Aufmerksamkeit gewidmet. Die erstellte Software ist dazu entwickelt, ein aufgenommenes Bild zu verarbeiten. Ebenso ermöglicht es die Software, das aufgenommene Bild unter Zuhilfenahme des vorher ausgeschnittenen Rahmens, vertikal wie auch horizontal auszurichten. Dazu müssen die acht schwarzen Punkte des Rahmens genau erfasst werden und deren Abstand sowie Position zueinander genau berechnet werden. Dies erfordert ein gewisses Maß an Erfahrung. Im Moment der Aufnahme benötigt es eine manuelle Korrektur des Bildes. Ungünstige Schattierungen und eine zu helle oder zu dunkle Aufnahme können die Detektion der Kreise so weit erschweren, dass die Aufnahme unbrauchbar ist. Ebenso ist darauf zu achten, dass die aufgenommene Fläche möglichst frei von störenden Motiven ist. Ein Schriftzug auf dem Motiv oder unsauber entsorgte Kreise

vom Schnitt können dafür sorgen, dass zu viele oder nicht die richtigen Kreise erkannt werden. Im diesem Fall benötigt es eine manuelle Korrektur der Aufnahme.
Sind die Kreise sauber erkannt, kann die Software selbstständig das Bild ausrichten und es liegt ein planes Foto von der Papierebene vor. Wie schon erwähnt, ist dies ein elementarer Bestandteil dieser Arbeit und das Erreichen dieses Zieles ist als Erfolg zu werten. Trotzdem ist nach reichlicher Überlegung die Erkenntnis gewachsen, dass ein anderes Muster für die Ausrichtung des Bildes unter Umständen besser geeignet wäre. Die Schwäche des Musters mit den acht Kreisen liegt darin, dass die Kanten 'ABCD' (vergleiche Abbildung: Skizze vom Schittmuster) rein virtuelle Kanten sind, die vom Lasercutter nicht angefahren werden. Dies hat den Nachteil, dass es keinen fixen Ankerpunkt für den zu schneidenden Pfad gibt. Das hier praktizierte 'Workaround' sieht vor, dass das Quadrat 'ABCD' vom Lasercutter mit minimaler Leistung ausgeschnitten wird, sodass die Punkte 'ABCD' im G-Code des Musters vorhanden sind, aber in der Abbildung nicht vorkommen. Ein Muster, in denen die Bezugspunkte real vorkommen, hätte diesen zusätzlichen Schritt nicht notwendig gemacht. Dennoch sind die Stärken des angewendeten Musters gut genug ausgeprägt, dass es für die Funktion der Bildausrichtung ausreichend genau ist.

4.3 Kanntenerkennung

Die Kantenerkennung hat hier überraschenderweise keine so tragende Rolle bei der Lösung der Aufgabe gespielt. Es ist jedoch wichtig zu beachten, dass das Motiv einen hohen Kontrast aufweist und sich deutlich vom Hintergrund abhebt. Falls dies nicht der Fall ist, sollte dieser Zustand herbeigeführt werden, indem das Motiv beispielsweise durch einen Farbfilter verstärkt oder abgeschwächt wird. Eine präzise Filterung störender Elemente ist ebenfalls wichtig. Im Idealfall gibt es ein Motiv, das als weiße Fläche dargestellt wird, und einen Hintergrund, der als schwarze Fläche dargestellt wird. Falls dies nicht durch Filter erzeugt werden kann, empfiehlt es sich, ein Polynom um das auszuschneidende Motiv zu zeichnen und das Innere des Polynoms als weiß und das Äußere als schwarz zu definieren.

4.4 Definition der Schnittkannte

Durch Einsatz der Simulation lässt sich die Schnittkante sehr leicht in ihrer Qualität überprüfen.

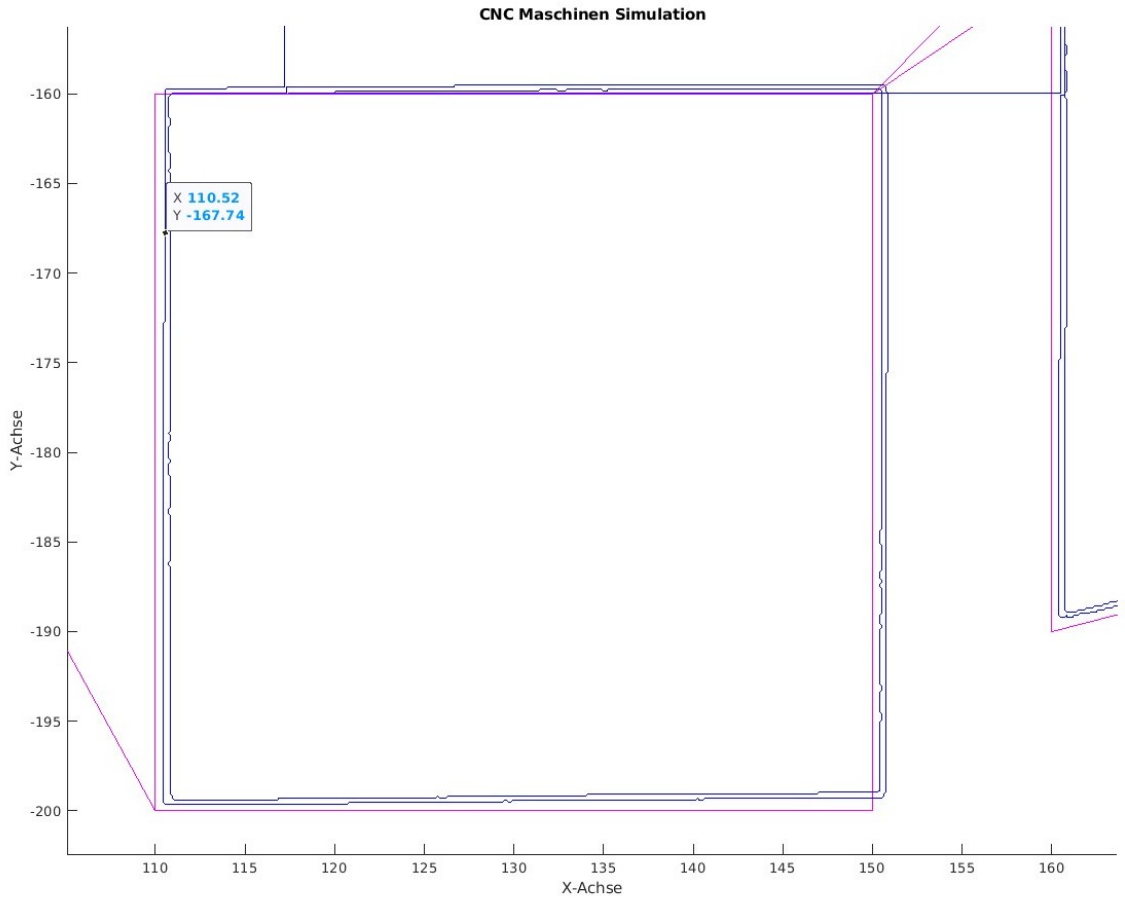

Abbildung 26: Überprüfung der Schnittkante

Die Abbildung 'Überprüfung der Schnittkante' zeigt das auszuschneidende Quadrat. Die magentafarbene Linie stellt das Original dar und die blaue Linie repräsentiert die Kante, die aus dem Bild erstellt wurde. Es ist deutlich zu erkennen, dass es nur eine geringe Abweichung zwischen den beiden Linien gibt.

Die absolute Abweichung in mm lässt sich an dieser Stelle ebenfalls leicht ablesen. Dafür müssen einfach die Koordinaten abgelesen werden, an denen die blaue Linie gezeichnet wurde. Da es sich hierbei um eine Darstellung eines G-Codes handelt und dieser G-Code mit der Einheit mm verwendet wurde, ist auch die Skala dieser Grafik als eine Skala in

mm zu interpretieren. Das originale Quadrat hat eine senkrechte Seite zwischen den Punkten X 110 / Y - 200 und X 110 / Y -160. Diese Seite ist genau 40 mm lang. Eine Stelle, an der der Abstand zwischen Ist und Soll groß ist, wurde von mir als Markierung in die Grafik aufgenommen. Es ist zu erkennen, dass sich der Punkt bei X110,52 / Y -167,74 befindet. Dies zeigt, dass die Anforderung, dass das System auf ein halbes mm genau ausschneiden soll, in der Simulation um 0,02 mm verfehlt wurde. Wenn man die absolute Ungenauigkeit des LC hinzurechnet, ergibt sich eine Gesamtungenauigkeit von 0,53 mm.

4.5 Fazit

Diese Arbeit stellt dar, wie versucht wurde, ein System zum Ausschneiden geometrischer Figuren anhand optischer Erfassung zu entwickeln. Zu Beginn wurden die technischen Grundlagen sowie bereits bestehende Techniken erläutert. Anschließend wurden Konzepte und Ideen vorgestellt.

Die Umsetzung und Realisierung erfolgte daraufhin, wobei die einzelnen Entwicklungsschritte detailliert dokumentiert und erklärt wurden. Abschließend fand eine Diskussion der Ergebnisse statt, in der kritisch reflektiert wurde, inwiefern die einzelnen Arbeitsschritte sowie das entwickelte System insgesamt erfolgreich waren.

Es gibt eine minimale Abweichung von der Vorgabe, dass das System eine Genauigkeit von 0,5 mm beim Ausschneiden erreichen soll. Diese Abweichung kann durch eine verbesserte Filterung und die Auswahl der richtigen Kantenerkennung weiter minimiert werden.

Die Arbeit zeigt, dass ich in der Lage bin, realistische Ziele zu setzen und den Willen und die Ausdauer habe, diese Ziele zu erreichen.

Danksagung

'Sie stehen auf den Schultern von Riesen', sagte Dr. Edzard Höfig zu mir, als er mir verständlich machen wollte, dass ich mir nicht jede Methode selbst ausdenken muss, sondern dass ich auch nachschlagen sollte, wie es andere Menschen machen. Dies ist sicher richtig. Doch die "Riesen" mit denen ich mich verbunden fühle und die wirklich ,im direkten Bezug, mich aktiv zum Vollenden dieser Arbeit getragen haben sprechen nicht über Lehrbücher zu mir, sondern meist über einen WhatsApp Chat.

In Allgemeinen möchte ich meiner Familie und meinen Freunden danken, die mich über die Zeit des Schreibens dieser Arbeit finanziell, materiell, emotional, aber auch durch konstruktive Kritik und durch das Korrekturlesen dieser Arbeit unterstützt haben. Im Speziellen danke ich meiner Tochter Ronja für ihre verständnisvolle Rücksicht, wenn ich keine Energie mehr hatte und gleichzeitig für das Schenken eines Energieschubs, um doch noch mit ihr raus zu gehen und mir zu zeigen, dass es wichtigeres gibt, als diese Arbeit.

Ich danke außerdem Julia, Karoline Wodara, Tilman Goebel, Fabian Wollny, Friedemann Fiß, Angela Frank, Thomas Schenk, Paula Timmel, Jadwiga und Adam Wodara für Essen, Tabak, Geld und ein offenes Ohr für meine Sorgen, Probleme und Gemecker, wenn es mal wieder nicht so gelaufen ist, wie ich es mir gewünscht habe.

Ihr seid die wahren Größen, die diese Arbeit erst möglich gemacht haben.

Danke.

Eigenständigkeitserklärung

Ich versichere, dass ich die Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen entnommen sind, sind als solche kenntlich gemacht. Die schriftliche und die elektronische Form der Arbeit stimmen überein. Ich stimme der Überprüfung der Arbeit durch eine Plagiatssoftware zu.

Ort, Datum Unterschrift

5 Literatur

Literatur

[1] Fetzer Fränkel. *Mathematik 1.* Springer Vieweg, WMXDesign GmbH, Heidelberg, 2012.

[2] Rüdiger Dillmann Pedram Azad, Tilo Gockel. *Computer Vision - Das Praxisbuch.* elektor, Elektor-Verlag GmbH, 52072 Aachen, 2009.

[3] Computer Vision Das Praxisbuch. *Vergeliche Seite 279 - 298.*

[4] Klaus Schörner. *Architekturfotografie: Technisch richtig und doch zu viel.* https://www.bonnescape.info/architekturfotografie-wieviel-shift-ist-richtig/.

[5] Dr. Reiner Steinbrecher. *Bildverarbeitung in der Praxis - Seite 137 Zeile 1 ff.* R. Oldenbourg Verlag Münschen Wien, R.Oldenbourg Verlag GmbH, München, 1993.

[6] Dr. Reiner Steinbrecher. *Bildverarbeitung in der Praxis - Seite 142 Kapitel 11.3.* R. Oldenbourg Verlag Münschen Wien, R.Oldenbourg Verlag GmbH, München, 1993.

[7] Sven Rens Anika Kehrer, Teja Philipp. *Lasercutting - Eigene Designs erstellen, schneiden gravieren.* Carl Hanser Verlag München, www. hanser-fachbuch.de, 2017.

[8] Sven Rens Anika Kehrer, Teja Philipp. *Lasercutting - Eigene Designs erstellen, schneiden gravieren Seite 95 letzte Zeile.* Carl Hanser Verlag München, www. hanser-fachbuch.de, 2017.

[9] skyforce. *Die Inspirationsquelle für diesen Code kommt vom User 'skyforce' auf der Seite https://www.gomatlab.de/kreise-im-bild-finden-forschleifen-vermeiden-t26346.html.*

[10] Konstantinos Fragkakis. *Die Inspirationsquelle für diesen Code kommt von https://de.mathworks.com/matlabcentral/answers/100813-how-do-i-find-the-indices-of-the-maximum-or-minimum-value-of-my-matrix.*

6 Anhang

Hier im Anhang finden Sie ein Speichermedium, welches das geschriebene Programm und alle zugehörigen Daten beinhaltet.