Berliner Hochschule für Technik

Entwicklung eines Algorithmus zur Echtzeit-Erkennung von Strichzeichnungen

Bachelorarbeit

Elias Jonathan Mannherz
921561
Humanoide Robotik - FB VII

16. September 2024 - 16. Dezember 2024

Betreuer:
Prof. Dr. Manfred Hild
M.Sc. Simon Untergasser

Zusammenfassung

Die vorliegende Arbeit bildet eine Grundlage für die Echtzeit-Erkennung von Strichzeichnungen durch den humanoiden Roboter Myon. Es wird ein Bildverarbeitungsalgorithmus zur Linienerkennung vorgestellt, welcher abstrakte Linien aus dem vorliegenden Kamerabild extrahiert und von der Funktionsweise des primären visuellen Kortex inspiriert wurde. Das Konzept der *Saliency Map* wird verwendet, um die Kanten relevanter Bildinhalte vom Hintergrund zu trennen; anschließend werden die vorhandenen Orientierungen durch morphologische Operationen isoliert und rekursiv zu abstrakten Linienobjekten umgewandelt. Die Implementierung einer bereits existierenden, lernfähigen Symbolerkennung war vorgesehen, wurde in dieser Arbeit jedoch nicht realisiert. Mit einer noch nicht optimierten, experimentell festgestellten Laufzeit von maximal 5 ms ist die Linienerkennung echtzeitfähig. Es werden Ansatzpunkte zukünftiger Verbesserungen sowie Möglichkeiten zur Weiterentwicklung aufgezeigt.

Generative oder GPT-basierte KI-Tools wurden nicht verwendet - weder in der Entwicklung des im Folgenden präsentierten Algorithmus noch in der zugehörigen Recherche. Gleiches gilt für das Verfassen dieser Bachelorarbeit, mögliche Revisionen sowie verwendetes Bildmaterial.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Kontext - Der humanoide Roboter Myon 2
1.2 Überblick über die frühen Verarbeitungsschritte im menschlichen Sehen . . . 5

2 Stand der Forschung **10**
2.1 Biologisch inspirierte Ansätze zur visuellen Verarbeitung 10
2.2 Probleme bei der Verwendung von Machine Learning für die visuelle Verarbeitung in robotischen Systemen 10
2.3 Algorithmen zur Erkennung handschriftlicher Linien und Symbole 11

3 Entwicklung des Algorithmus **14**
3.1 Präsentation des entwickelten Algorithmus zur Extraktion von Linien 16
3.2 Perspektiven für die Weiterentwicklung 24

4 Ergebnisse **26**
4.1 Linienerkennung in Buchstaben aus dem MNIST-Datensatz 26
4.2 Linienerkennung in Strichzeichnungen 29
4.3 Linienerkennung in elektrischen Schaltplänen 33
4.4 Erkennbarkeit der re-synthetisierten Strichzeichnungen 35
4.5 Laufzeit 39
4.6 Vergleich des implementierten Algorithmus zum menschlichen Sehen 40

5 Fazit **41**

A Glossar **42**
A.1 Roboter 42
A.2 Wahrnehmung von Farben 43
A.3 Morphologische Operationen 44
A.4 Nachbarschaft 46
A.5 Neuronales Netz 47

B Erweiternde Abbildungen **48**

Literatur **52**

Abbildungsverzeichnis **54**

1 Einleitung

Sehen ist der dominanteste Sinn, den wir als Menschen haben.
Sehen befähigt uns sowohl dazu, weit entfernte Objekte zu erkennen und deren Distanz zu uns einzuschätzen, als auch dazu, nahe Objekte sehr detailliert zu untersuchen, ohne sie berühren zu müssen.
Sehen erlaubt es uns, mittels Schrift und Bild Informationen an andere Personen zu übermitteln, die weit entfernt oder sogar in späteren Zeiten leben, und mit nur einem Text beliebig viele Menschen zu erreichen.

Tatsächlich werden laut Man und Olchawa (2018) [1] ungefähr 80 % der uns zur Verfügung stehenden Informationen über unsere Umwelt durch das Sehen gewonnen. Es ist daher naheliegend, dass die Befähigung zur Sicht (im technischen Kontext: die visuelle Verarbeitung) auch für die Robotik essenziell ist. In dieser Arbeit wird der Versuch dokumentiert, die Fähigkeit des Sehens in dem humanoiden Roboter Myon zu erweitern. Konkret soll er dazu befähigt werden, einfache Symbole oder Strichzeichnungen zu erkennen, wobei mit Strichzeichnung eine Zeichnung aus einem oder mehreren Strichen gemeint ist.

Es soll sich um eine Erkennung in Echtzeit handeln, was bedeutet, dass eine für die Erkennung zugewiesene Zeitspanne verlässlich nicht überschritten wird. Während heutige, käuflich erwerbbare humanoide Roboter meist die Verarbeitung von Stimmen oder Bildern auslagern, indem sie ihre Daten zur Analyse an einen Server schicken, soll die hier entwickelte Erkennung nicht nur in Echtzeit, sondern auch lokal erfolgen. Dies ist zum einen zwingend erforderlich, da Myon mit diesem Prinzip entwickelt wurde, es hat jedoch auch klare Vorteile: Neben reduzierten Latenzzeiten ist insbesondere die Möglichkeit zu nennen, aktiv fehlende Informationen einzuholen und so in eine Interaktion mit der Umgebung zu treten: Ist bspw. ein zu erkennendes Symbol im Bild zu klein, um es klar zu erkennen, so kann darauf zugegangen werden und es muss nicht geraten werden. Während dieses Konzept des „aktiven Sehens“ (engl. *active vision*) in dieser Arbeit nicht explizit verwendet wird, ist es für die Philosophie in der Arbeit mit Myon sehr relevant: Es muss kein langsames System entwickelt werden, welches auf Basis von nur einem Bild fast immer perfekt akkurate Resultate liefert - ein schnelles und einfaches System, mit dem über mehrere Bilder ein ähnlicher Effekt erzielt werden kann, ist meist besser, da solche Systeme typischerweise weniger störanfällig und flexibler sind.

Die Arbeit ist wie folgt strukturiert: Zunächst wird kurz auf den Kontext der Arbeit eingegangen, insbesondere wird Myon vorgestellt, zusammen mit einer Erklärung spezifischer Begriffe (Abschnitt 1.1). Anschließend wird eine kurze Erläuterung des aktuellen Wissensstands über den Prozess des menschlichen Sehens gegeben (Abs. 1.2). Damit soll zum einen eine Verständnisgrundlage für die inhärente, uns jedoch alltäglich erscheinende, Komplexität dieses Prozesses geschaffen werden und zum anderen sowohl das Konzept als auch die Motivation für die spätere Verwendung der Wichtigkeits-Karte (engl. *Saliency Map*) eingeführt werden. Danach wird ein Überblick über den aktuellen Forschungsstand im Feld der Handschrift- und Zeichnungserkennung gegeben, mit besonderem Fokus auf für diese Arbeit relevante Konzepte (Abs. 2).

Es folgt die Vorstellung des entwickelten Algorithmus (Abs. 3.1) sowie die Motivation für in der Entwicklung getroffene Entscheidungen (Abs. 3). Da aus zeitlichen Gründen nicht alle Ziele erreicht wurden, wird außerdem ein kurzer Ausblick auf die noch geplanten Entwicklungen gegeben (Abs. 3.2).

Im letzten Teil der Arbeit wird an Hand von Beispielen die aktuelle Funktionalität präsentiert und bewertet (Abs. 4), gefolgt von einem kurzen Fazit (Abs. 5). Verwendete Konzepte, welche nicht Myon-spezifisch sind und daher nicht in Abs. 1.1 erklärt wurden, werden entweder bei Verwendung oder im Glossar eingeführt; dieses kann bei Bedarf referenziert werden (Appendix A).

1.1 Kontext - Der humanoide Roboter Myon

Die vorliegende Bachelorarbeit wurde vor dem Hintergrund des Projekts „RoSen-JunKer“ angefertigt. RoSen-JunKer steht für „Humanoide Robotik mit Senioren, Jugendlichen und Kindern“ und wird zum Zeitpunkt dieser Arbeit vom Bundesministerium für Bildung und Forschung (BMBF) im Rahmen der DATIpilot Förderrichtlinie unterstützt. Es handelt sich um eine Weiterführung früherer Projekte.

Eine zentrale Idee des Projektes ist es, der Öffentlichkeit die Möglichkeit zur Interaktion mit einem humanoiden Roboter zu geben. Bei dieser Interaktion sollen beide Seiten profitieren: Die Besuchenden haben kostenlos die Möglichkeit, direkt mit einem humanoiden Roboter zu interagieren und dabei einen Einblick in das Forschungsfeld der Humanoiden Robotik zu erhalten, während die Forschenden direktes Feedback über die Wünsche und Eindrücke von Personen mit verschiedensten fachlichen Hintergründen erhalten, welches dann in zukünftige Entwicklungen einfließen kann.

Zum Zeitpunkt dieser Arbeit erfolgen diese Interaktionen noch hauptsächlich auf Basis von (akustischer) Spracherkennung. Das Ergebnis dieser Arbeit soll dieses Repertoire erweitern und die Türen zu weiteren Interaktionen und Fähigkeiten öffnen.

Myon, zu sehen in Abbildung 1, ist der humanoide Roboter, mit dem diese Interaktionen stattfinden. Er wurde vom Forschungslabor Neurorobotik (NRL) der Berliner Hochschule für Technik (BHT) designt und gebaut und wird dort fortlaufend weiterentwickelt.

Abbildung 1: Der humanoide Roboter Myon[1]

Die Vision des NRL ist es, dass Myon als lernfähiges Teammitglied wahrgenommen wird. Dies lässt sich im Design erkennen: Myon ist weder groß noch stark - seine Größe und Proportionen entsprechen eher denen eines sieben Jahre alten Kindes. Es handelt sich nicht um einen Roboter, welcher Menschen die Arbeit abnehmen soll - es geht um die Interaktion mit Menschen, weshalb er eher zugänglich und leicht tapsig aussehend designt wurde.

1.1.1 PixelPipeline

Myon ist mit einer Bild-Vorverarbeitungspipeline ausgestattet, der PixelPipeline. Die Vorverarbeitung durch die PixelPipeline erfolgt mittels eines FPGA-Moduls (*„Field Programmable Gate Array“*) und beansprucht damit nahezu keine Prozessorleistung.

[1] Bildquelle: `http://www.neurorobotik.de/img/robots/myon/myon_outside.jpg`

Einer der Vorverarbeitungsschritte der PixelPipeline ist es, die eingehenden Daten der Kamera zu komprimieren. Dabei werden Pixel des eingehenden Kamerabildes zu Superpixeln (vgl. Abs. 1.1.2) kombiniert; die häufigste Verwendung dieses Features ist es, jeweils einen quadratischen Bereich des Bildes zu einem Superpixel zusammenzufassen, sodass das gesamte Bild abschließend mit reduzierter Auflösung vorliegt. Das resultierende Superpixel erhält dabei den durchschnittlichen Farbwert aller beteiligten Pixel.

Im Rahmen dieser Arbeit wird auf Basis eines in dieser Weise komprimierten Bildes gearbeitet, welches 22×22 Superpixel groß ist. Das aus der Vorverarbeitung resultierende Bild wird in einem auf 4 kB begrenzten Speicherbereich abgelegt, wobei hier ein Superpixel 8 Byte einnimmt. Damit ist ein Maximum von 512 Superpixeln pro Bild möglich, weshalb 22×22 Superpixel als Bilddimensionen gewählt wurden; dies resultiert in 484 Superpixel pro Bild, während bei 23×23 Superpixeln stattdessen 529 Superpixel pro Bild nötig wären - es wurde also die größtmögliche Kantenlänge für ein quadratisches Bild gewählt. Nicht-quadratische Bildformate sind ebenfalls möglich, finden in dieser Arbeit jedoch keine Verwendung.

Die Anzahl der Pixel pro Superpixel kann ebenfalls variiert werden; auch hier fiel die Entscheidung auf eine quadratische Form. Für die gewählten Dimensionen ist der sinnvolle Wertebereich für die Anzahl der beteiligten Pixel pro Superpixel $[1^2, 13^2]$ ($[1, 169]$), da das eingehende Kamerabild in einer Auflösung von 360×288 Pixeln vorliegt und mit 22 Superpixeln je 13 Pixel eine Höhe von 286 Pixeln im Kamerabild vorausgesetzt wird.

Wird die Anzahl der Pixel pro Superpixel reduziert, so resultiert dies in einer Vergrößerung des ausgewählten Bildausschnitts, da die Dimensionen des Bildes (22×22) erhalten bleiben, unabhängig davon, wie viele Pixel Teil eines Superpixels sind. Es ist also ein 13-stufiger Zoom möglich. Da im Zuge der Vorverarbeitung durch die PixelPipeline eine 2×3-Transformationsmatrix auf das Bild angewandt wird, ist es u.a. möglich, den betrachteten Bildbereich zu verschieben, sodass bei Verwendung des Zooms ein ausgewählter Bildbereich fokussiert werden kann.

Ein neues Bild liegt alle 20 ms vor und die Farbtiefe ist 8 Bit. Das verwendete Farbmodell ist das YUV-Farbmodell (vgl. Abs. 1.1.3).

1.1.2 Superpixel

Als „Superpixel“ wird im Folgenden eine Gruppe von Pixeln verstanden, welche irreversibel, also mit Informationsverlust, zusammengefasst werden. Das Kamerabild von Myon liegt in Form solcher Superpixel vor, welche durch die Bildvorverarbeitung der PixelPipeline erzeugt werden (vgl. Abs. 1.1.1).

Den erzeugten Superpixeln wird als Farbwert jeweils das arithmetische Mittel aller an diesem Superpixel beteiligten Pixel zugewiesen. Die Regel, mittels derer die Pixel zu Superpixeln gruppiert werden, kann innerhalb der PixelPipeline eingestellt werden; in dieser Arbeit werden quadratische Bildbereiche zu Superpixeln zusammengefasst.

Ein einfaches Beispiel für die Bildung von Superpixeln ist in Abb. 2 dargestellt:

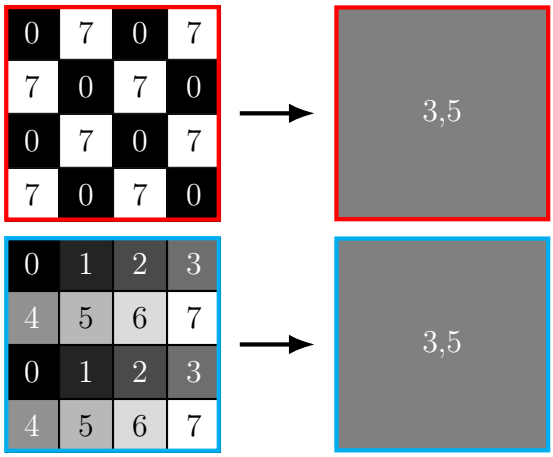


Abbildung 2: Kombination von Pixeln zu Superpixeln, hier mit Graustufen im Wertebereich $[0, 7]$. Es werden immer 16 Pixel (links farbig umrandet) zu dem entsprechend umrandeten Superpixel (rechts) zusammengefasst.

Das im Bild mit Cyan umrandete Superpixel, bei dem in den beteiligten Pixeln jeder mögliche Farbwert zweimal vorkommt, erhält als Farbwert die Mitte des möglichen Wertebereichs, was hier einem mittleren Grau entspricht.

Dies ist die gleiche Farbe, die das mit Rot umrandete Superpixel erhält, bei dem die Hälfte der beteiligten Pixel weiß und die andere Hälfte schwarz ist. Die beiden Superpixel lassen sich nicht mehr unterscheiden, obwohl die zugrunde liegenden Pixel sehr verschieden waren; dies zeigt den genannten Informationsverlust.

1.1.3 YUV-Farbmodell

Die bekannteste Repräsentation von Farben in einem digitalen Kontext ist RGB: Eine beliebige Farbe wird als Kombination aus Rot-, Grün- und Blau-Anteilen dargestellt und alle möglichen RGB-Kombinationen bilden gemeinsam einen dreidimensionalen Farbraum.

Dabei entsteht eine Hauptdiagonale von $(0, 0, 0)$ zu $(255, 255, 255)$, entlang derer sich die Helligkeit verändert.

Eine alternative Darstellung ist YUV: Hier wird stattdessen auf der Y-Achse die Helligkeit abgebildet, während auf der U-V-Ebene der Farbton codiert wird. Verschiedene Repräsentationen haben unterschiedliche Vor- und Nachteile, bspw. ist ein Vorteil der YUV-Darstellung, dass sie dem menschlichen Sehen ähnlicher ist als die RGB-Darstellung.

Da bei Myon (vgl. Abs. 1.1) die YUV-Darstellung verwendet wird, ist im Folgenden mit „Farbraum“ immer der Farbraum im Sinne des YUV-Modells gemeint. Als „Farbwert“ wird entsprechend ein Punkt (y, u, v) in diesem Farbraum verstanden: Ein mittleres Grau würde bspw. als $(128, 128, 128)$ dargestellt (Abb. 3); die Kenntnis spezifischer Werte ist für diese Arbeit jedoch nicht von Belang.

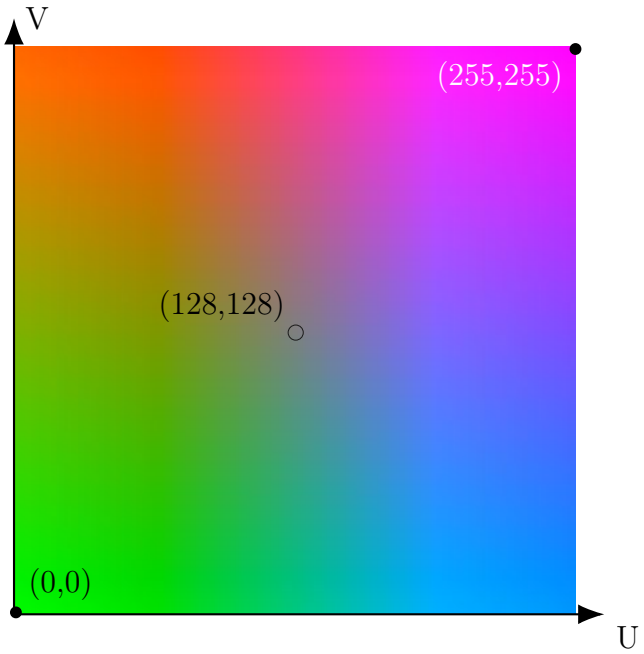


Abbildung 3: Eine Visualisierung der UV-Ebene bei konstantem Y-Wert: Y = 128.[2]

[2]Leicht bearbeitet. Autor des Originalbildes: Simon A. Eugster. Entnommen unter public domain license von `https://commons.wikimedia.org/wiki/File:YUV_UV_plane_Y0.5_50_percent.png`

1.2 Überblick über die frühen Verarbeitungsschritte im menschlichen Sehen

Wie viele andere Lebewesen sind wir Menschen in der Lage, aus dem uns erreichenden Licht Informationen zu gewinnen:

Ein Lichtstrahl trifft unser Auge. Auf dem Weg von der Lichtquelle zu unserem Auge kann sich das Spektrum des Lichtstrahls verändert haben, bspw. durch eine partielle Reflexion des Lichtstrahls an einem Objekt oder durch die Brechung des Lichtstrahls in einem Medium wie der Atmosphäre. Auf diese Weise sind in dem Lichtstrahl Informationen über den zurückgelegten Pfad enthalten, was es uns ermöglicht, auf Basis dieser Informationen Rückschlüsse auf unsere Umgebung zu ziehen; dieser Prozess wird im Allgemeinen als „Sehen“ bezeichnet.[3]

Die genaue Funktionsweise des menschlichen Sehens ist ein aktives Forschungsfeld und viele Fragen sind noch ungeklärt. Tatsächlich sind selbst die frühen am Sehprozess beteiligten Systeme so komplex, dass es laut Benoit et al. (2010) [3] in der Retina noch immer Zelltypen gibt, deren Funktion unklar ist. Dieses Kapitel soll dennoch einen groben Überblick darüber geben, was in den frühen Schritten der menschlichen visuellen Verarbeitung passiert.

1.2.1 Linse

Licht, welches das menschliche Auge trifft, kann nur durch die Pupille in das Auge gelangen. Die Pupille liegt im Zentrum der Iris und ihre Größe kann u.a. in Abhängigkeit der Umgebungshelligkeit verändert werden, um die Menge des einfallenden Lichts zu modulieren. Die Iris hat dabei eine ähnliche Funktion wie die Blende einer Kamera. Fällt ein Lichtstrahl in ein menschliches Auge, so wird er von der Linse auf einen Bereich der Netzhaut gebrochen, welcher sowohl vom Einfallswinkel des Lichtstrahls durch die Pupille als auch von der lokalen Dicke der Linse abhängt. Der von außen sichtbare Bereich des Auges fungiert also als eine Abbildungsfunktion des gesamten Sichtfeldes auf das Innere des Auges (genauer: auf die Retina).

Bei dieser Abbildung kommt es zu einer Vorzeichenumkehr der Bildkoordinaten, sodass Objekte aus der linken Hälfte des Sichtfeldes auf die rechte Hälfte der Retina und Objekte aus der oberen Hälfte des Sichtfeldes auf die untere Hälfte der Retina abgebildet werden. Diese Vorzeichenumkehr erfolgt in beiden Augen gleichermaßen, sodass ein Objekt auf beinahe die gleiche Position in beiden Retinae abgebildet wird.

Ob das Objekt auf exakt die gleiche Position abgebildet wird, hängt von mehreren Faktoren ab, von denen manche verändert werden können (bspw. die Orientierung der Augen zueinander) und manche fest sind (bspw. der Abstand der Augen zueinander). Wird ein Objekt in beiden Augen auf die gleiche Position abgebildet, so wird dies als „das Objekt scharf sehen“ wahrgenommen; aus den dafür nötigen Parametern wird später die Distanz zu dem Objekt errechnet. Um die Distanz auf diese Weise zu messen, ist es nicht nötig, dass das Objekt scharf gesehen wird, da im visuellen Kortex die Positionsdifferenz der beiden Abbildungen in die Berechnung einfließt.

1.2.2 Retina

Die Retina, auch Regenbogenhaut, ist eine dünne Zellschicht, welche die der Pupille gegenüberliegende Innenwand des Auges bedeckt. Die Retina enthält die Photorezeptorzellen des Auges, also diejenigen Zellen in der menschlichen visuellen Wahrnehmung, die auf das einfallende Licht reagieren.

[3] Strukturelles Zitat aus [2]

Die Retina ist ungefähr rotationssymmetrisch aufgebaut:[4] Im Zentrum liegt die sog. Fovea Centralis. Dabei handelt es sich um einen kleinen Bereich mit ca. 1.5 mm Durchmesser, in dem eine extreme Dichte an Photorezeptorzellen vorliegt. Dies ist daher der am höchsten auflösende Bereich der Retina. Die meisten Photorezeptorzellen in der Fovea Centralis sind sog. Zapfenzellen, welche mit Farbwahrnehmung assoziiert werden. Im menschlichen Auge sind drei verschiedene Typen Zapfenzellen vertreten. Sie unterscheiden sich durch die vermehrte Präsenz jeweils eines Moleküls, welches auf einen gewissen Bereich des Optischen Spektrums reagiert; tatsächlich wird das Optische Spektrum darüber definiert, welche Anteile des elektromagnetischen Spektrums (also: des Lichts) von menschlichen Photorezeptorzellen wahrgenommen werden können. Die drei Varianten der Zapfenzelle haben also drei verschiedene spektrale Absorptionskurven, was uns zusammengenommen das Sehen von Farbe ermöglicht.

Während in der Fovea Centralis fast nur Zapfenzellen vorkommen, nimmt ihre Anzahl mit zunehmendem Abstand zur Fovea Centralis drastisch ab. Im restlichen Bereich der Retina überwiegt die andere Sorte Photorezeptorzelle: die sog. Stäbchenzelle. Alle Stäbchenzellen haben die gleiche spektrale Absorptionskurve, weshalb sie nicht für die Farbwahrnehmung, sondern nur für die Wahrnehmung von Helligkeit verwendet werden. Die Häufigkeit dieser Stäbchenzellen nimmt zum Rand der Retina ebenfalls ab, allerdings ist der Abfall sehr viel flacher als im Falle der Zapfenzellen. Dies resultiert darin, dass die Farbwahrnehmung zum Rand des Sichtfeldes sehr viel schneller abfällt als die Wahrnehmung von Helligkeitsdifferenzen. Stäbchenzellen sind sensibler als Zapfenzellen für geringe Mengen an Licht, weshalb das Gesehene in dunkleren Lichtverhältnissen zunehmend monochrom erscheint. Sie sind ebenfalls mit der Wahrnehmung von Bewegung assoziiert, weshalb eine Bewegung am Rande des Sichtfeldes sehr viel klarer wahrgenommen werden kann als eine farbliche Veränderung.

Trifft ein Lichtstrahl also auf einen Bereich der Retina, so werden die in diesem Bereich vorhandenen Photorezeptorzellen aktiviert und die Stärke ihrer Aktivierung hängt von dem Spektrum des Lichtstrahls und dem Typ der vorhandenen Photorezeptorzellen ab.

Über den Photorezeptorzellen liegen zwei weitere Zellschichten.[5] Die Axone der oberen Schicht, welche aus Ganglienzellen besteht, bilden gemeinsam den Sehnerv, welcher die Signale der „rezeptiven Felder“ ins Gehirn weiterleitet. Bei einem rezeptiven Feld handelt es sich um eine Zusammenfassung der Aktivität mehrerer „benachbarter“ Photorezeptorzellen (für den Begriff der Nachbarschaft siehe App. A.4).

Während rezeptive Felder in verschiedenen Größen vorkommen, lassen sie sich in zwei Gruppen unterteilen: „ON-Center“ und „OFF-Center“. In beiden Fällen wird die Aktivität der Photorezeptorzellen im zentralen Kern-Bereich des rezeptiven Feldes mit der im äußeren Rand-Bereich verglichen. Ein ON-Center rezeptives Feld reagiert stark darauf, wenn die Photorezeptorzellen im Kern-Bereich des rezeptiven Felds aktiviert werden *und* jene im Rand-Bereich nicht aktiv oder nur leicht aktiv sind. Ein OFF-Center rezeptives Feld verhält sich invers dazu. Beide Varianten sind nicht aktiv, wenn keine Signalaktivität herrscht, und schwach aktiv, wenn Zellen in beiden Bereichen ähnlich aktiv sind.

[4] Der blinde Fleck ist hier nicht von Bedeutung und wird ignoriert.

[5] „Über“ meint hier, dass sie zwischen der Pupille und den Photorezeptorzellen liegen, allerdings nur zu den Photorezeptorzellen „benachbart“ sind.

Die Aktivität der rezeptiven Felder wird über den Sehnerv, durch das Chiasma Opticum an die seitlichen Kniehöcker und dann an den primären visuellen Cortex (V1) übermittelt. (Dieser Pfad ist in Abb. 4 dargestellt; es wird in folgenden Abschnitten kurz auf die genannten Hirnareale eingegangen.)

Ist das Signal im V1 angekommen, so propagiert es dann weiter in die übrigen Bereiche des visuellen Kortex und weitere Hirnregionen. Je weiter sich das Signal entlang dieses Pfades von der Retina entfernt, desto unklarer wird es - bei aktuellem Wissensstand - was genau damit geschieht, da die Vernetzung zu anderen Bereichen des Gehirns zu einem hohen Ausmaß an Interkonnektivität und damit Komplexität führt.

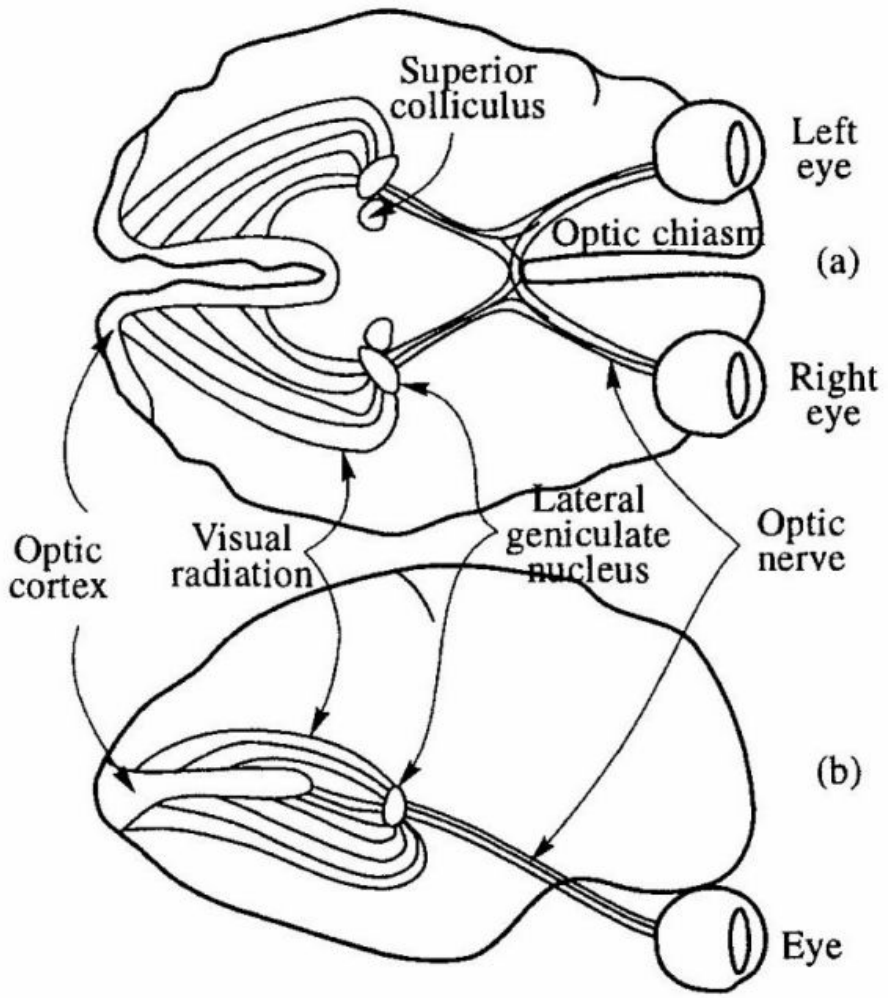


Abbildung 4: Darstellung der Nervenbahnen von den Augen zum (primären) visuellen Kortex beim Menschen, entnommen aus [4]. Dargestellt sind - von rechts nach links gehend - die Augen, das Chiasma Opticum (*Optic chiasm*, nur im oberen Bildteil), die seitlichen Kniehöcker (*Lateral geniculate nucleus*) und der visuelle Kortex (*Optic cortex*), neben weiteren Details. Der *primäre* visuelle Kortex (V1) ist Teil des visuellen Kortex; es ist ein kleiner Bereich, mittig und ganz hinten im Großhirn (ganz links im Bild), ungefähr dort verortet, wo die Pfeilspitzen des „*Optic cortex*“-Labels enden; der V1 ist Teil beider Hemisphären.

1.2.3 Chiasma Opticum

Um aus den noch unabhängigen „Bildern“ beider Augen ein zusammengehöriges Ganzes zu erzeugen, beginnt im sog. *Chiasma Opticum* (griechisch, ungefähr: „Kreuzung des Sehens“) der Prozess, die beiden „Bilder“ zusammenzufügen: Beide Sehnerven (welche jeweils aus vielen Axonen bestehen) werden hier so aufgeteilt, dass anschließend die Signale der beiden *linken* Hälften der Retinae getrennt von den Signalen der beiden *rechten* Hälften vorliegen.

Diese Kombination der beiden „Bilder“ ist zu diesem Zeitpunkt noch nicht vollständig abgeschlossen: Zhaoping (2019) [5] beschreibt Testergebnisse, die zeigen, dass die beiden Bilder im primären visuellen Kortex (V1) mindestens teilweise noch getrennt vorliegen und dass dies Auswirkungen auf die Blickführung haben kann; „nach“ dem V1 gibt es jedoch fast nur noch binokulare Neuronen, also solche, die auf Basis eines kombinierten Bildes arbeiten.

1.2.4 Seitlicher Kniehöcker - LGN

Die Signale der Retinae werden anschließend über die seitlichen Kniehöcker (LGN, „*Lateral Geniculate Nucleus*“) an den primären visuellen Kortex weitergegeben.

Mögliche zusätzliche Funktionen der LGN sind noch nicht abschließend erforscht. Dies liegt u.a. daran, dass die Retinae laut Krauskopf (1998) [4] jeweils nur ca. 10 % der in die LGN eingehenden Signale ausmachen. Die restlichen 90 % sind eingehende Signale von anderen Hirnarealen, was die Analyse erschwert; die Funktion dieser Signale ist noch fast gänzlich unbekannt.

1.2.5 Primärer visueller Kortex - V1

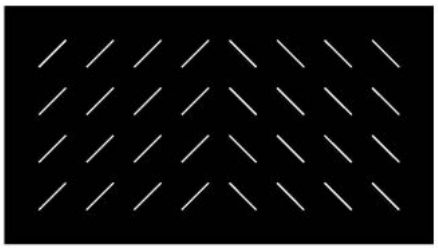
Zu dem Zeitpunkt, zu dem die Signale der Retina den primären visuellen Cortex (V1) erreichen, sind sie bereits stark komprimiert worden: Die Retina erhält ungefähr 100 MB/s an eingehenden Daten (Kelly, 1962, zitiert nach Zhaoping, 2019) [5], es werden jedoch „nur“ rund 1 MB/s für die weitere Analyse weitergegeben (Zhaoping, 2014, zitiert nach Zhaoping, 2019). Dieser Trend der Datenreduktion wird im V1 und folgenden Hirnarealen bis zu einer Datenrate von 40 Bit/s fortgesetzt, welche als der „Menschliche-Aufmerksamkeits-Engpass“ (engl. *Human attentional bottleneck*) bezeichnet wird (Kelly, 1962, sowie Sziklai, 1956, zitiert nach Zhaoping, 2019).

Die eingehende Informationsmenge wird in der Retina durch die rezeptiven Felder stark zusammengefasst - im V1 beginnt die weitere Kompression der Daten auf das Wichtigste im Bild. Dieses „Wichtigste“ lässt sich graphisch durch eine „Wichtigkeits-Karte“ (engl. *Saliency Map*) darstellen, welche nach aktuellem Verständnis im V1 erzeugt wird. Entscheidend für das aktuelle Verständnis davon, wie diese Wichtigkeits-Karte erzeugt wird, sind zwei Beobachtungen: Zum einen liegen alle Neuronen, die auf das gleiche „Merkmal“ (engl. *Feature*) im Bild reagieren, benachbart zueinander als Cluster vor. (Der Begriff der „Nachbarschaft“ wird in App. A.4 erläutert). Es gibt sehr viele solcher Neuronencluster im V1, welche jeweils auf unterschiedliche Merkmale reagieren.

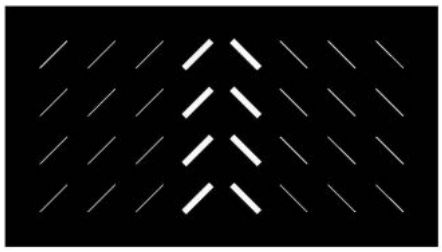
Tatsächlich lassen sich hier strukturelle Parallelen zwischen der visuellen Verarbeitung beim Menschen und der in künstlichen Neuronalen Netzen (NN, vgl. App. A.5) erkennen. Olah et al. (2020) [6] untersuchten die Strukturen größerer, auf visuelle Verarbeitung trainierter NNs, um die einzelnen Schritte innerhalb des NNs besser zu verstehen. Laut Olah et al. gab es in jedem nicht-trivialen Modell, das sie untersuchten, Neuronen, welche stark auf verschiedene Sorten Kurven reagierten. Dies ist eine direkte Parallele zur menschlichen visuellen Verarbeitung, da auch im visuellen Kortex solche Strukturen vorkommen.

Die andere für die Wichtigkeits-Karte relevante Beobachtung ist, dass ein aktives Neuron im V1 die Aktivierung der benachbarten Neuronen schwächt. Aus diesen beiden Beobachtungen resultiert, dass *jenen* Neuronen eine höhere Aktivierung möglich ist, welche die *Kanten* eines Bildbereiches als Eingangssignal erhalten. Das ist darin begründet, dass einige ihrer Nachbarn nicht aktiv werden - unter der Annahme, dass dieser Bildbereich sonst homogen ein dieses Neuronen-Cluster aktivierendes Merkmal aufweist, die Umgebung dieses Merkmal jedoch nicht aufweist. Die Aktivierung von Neuronen, welche vollständig von aktiven Neuronen umgeben sind, werden entsprechend stärker geschwächt.

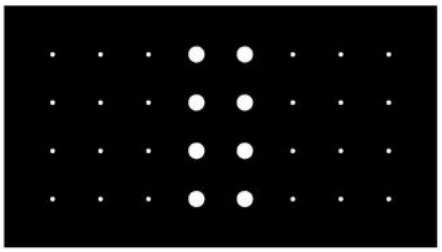
Dies bedeutet konkret, dass die *Kanten* eines Objekts im Bild einen höheren Wert in der Wichtigkeits-Karte erreichen als die *Flächen* des Objekts; die Verarbeitung im V1 lässt sich also als eine Art „Kantendetektion“ beschreiben. Die Merkmale, auf welche verschiedene Neuronen-Cluster im V1 reagieren, sind vielfältig: So wird nicht nur auf Helligkeit oder Farbe reagiert, durch mehrere aufeinander folgende Bearbeitungsschritte reagieren manche Neuronen-Cluster auch auf das Vorkommen simpler Formen oder deren Orientierung. Auf diese Weise können auch Kanten zwischen unterschiedlichen Orientierungen wahrgenommen werden, wie Abb. 5 zeigt. Dieses Modell für die Funktionsweise des V1 ist als *V1 Saliency Hypothesis* bekannt.



(a) Eingehendes Bild



(b) Aktivierung in V1



(c) Wichtigkeits-Karte

Abbildung 5: Schematische Darstellung der Wahrnehmung einer „Kante“ zwischen zwei „Flächen“ unterschiedlich orientierter Linien durch den V1. Es wird davon ausgegangen, dass der hier gezeigte Ausschnitt ein Teil eines größeren Bildes ist, weshalb keine Außenkanten der Bereiche wahrgenommen werden. Das hier Gezeigte ist ein Teil einer Grafik aus [7].

2 Stand der Forschung

In diesem Kapitel soll der aktuelle Forschungsstand im Bereich der Erkennung von handschriftlichen Zeichnungen und Symbolen dargestellt werden. Dabei wird besonderer Fokus auf für diese Arbeit relevante Veröffentlichungen gelegt.

Zunächst sei die Bachelorarbeit von Wend (2024) [8] genannt, welche ein ähnliches Thema bearbeitet hat wie die hier dokumentierte Arbeit und ebenfalls im NRL entstand, auch wenn nicht direkt mit Myon gearbeitet wurde. Der von ihm vorgestellte Ansatz wurde hier nicht weiter verfolgt, u.a. basierend auf der von ihm in seiner Arbeit formulierten Bewertung desselben.

In seiner Dissertation beschreibt Panreck (2021) [9] eine erfolgreiche Implementierung des Harris-Detektors zur Erkennung von Ecken und Kanten mit Myon; das Ziel war es, Myon Linien nachzeichnen zu lassen. Die von ihm beschriebene Methode wurde im Entwicklungsprozess dieser Arbeit ebenfalls verworfen, da sie zu aufwändig schien; die Validität dieser Bewertung wurde inzwischen allerdings in Frage gestellt und sollte überprüft werden.

2.1 Biologisch inspirierte Ansätze zur visuellen Verarbeitung

Eine möglicherweise naheliegende Idee wäre es, zu versuchen, die visuelle Verarbeitung beim Menschen nachzuahmen. Benoit et al. (2010) [3] beschreiben einen solchen Versuch: Sie modellieren die Funktionen der Retina und des V1 und erzielen nach eigener Aussage eine Neutralisierung von Rauschen und Hellgikeitsvarianz, „statische und dynamische Konturenverstärkung“ sowie eine frequenz- und orientierungsbasierte Analyse. Ihre Arbeit wurde mit Bildern in Graustufen durchgeführt, was die Autoren damit begründen, dass „auf der Ebene der Retina noch viele Zellen und deren Funktionen unbekannt sind“ (Übersetzung des Autors).

Ein tatsächlich umsetzbares Nachvollziehen des von Benoit et al. vorgestellten Algorithmus war dem Autor dieser Arbeit auf Basis der genannten Quelle jedoch nicht möglich.

2.2 Probleme bei der Verwendung von Machine Learning für die visuelle Verarbeitung in robotischen Systemen

Die meisten automatisierten Aufgaben, welche Bildverarbeitung beinhalten, erfordern die Kategorisierung der Bildinhalte. Zu einer solchen Kategorisierung gehört nicht nur die Erkennung gewisser Formen oder Symbole, sondern auch eine Entscheidung darüber, was in dem Bild relevant ist und wo die Grenzen zwischen einzelnen Objekten sowie zwischen Vorder- und Hintergrund liegen.

Dies wird heutzutage meist durch die Verwendung von Maschinellem Lernen (ML) und Neuronalen Netzen (NN) realisiert (für eine grundlegende Beschreibung von NNs siehe App. A.5). Zum Kategorisieren wird insbesondere Deep Learning (DL) häufig eingesetzt, bei welchem ein NN unter Zuhilfenahme von großen Datenmengen für eine spezifische Aufgabe trainiert wird.

Während ML in vielen Anwendungsfeldern gut funktioniert, ist diese Technik nicht optimal in Situationen, welche Lernfähigkeit erfordern. Ein Grund dafür ist, dass ML-Systeme i.A. in zwei Phasen existieren. Am Beispiel von DL:

Es gibt die Lernphase, in welcher das NN trainiert wird, bspw. darauf, Bilder zu kategorisieren. „Training“ bedeutet hier, dass die Gewichte des NNs immer weiter angepasst werden, bis das Ausgangssignal des NNs (im Beispiel: Die Kategorisierung des Bildes) den Anforderungen der gegebenen Aufgabenstellung entspricht. Sobald die Lernphase abgeschlossen ist, kann das System eingesetzt werden (Deployment-Phase).

In der Deployment-Phase sollte das System allerdings nicht mehr verändert werden. Wie in App. A.5 weiter erklärt wird, kann die Veränderung einzelner Gewichte einen großen Einfluss auf das Verhalten des gesamten Systems haben:

Kein einzelnes Gewicht „speichert“ eine spezifische Information, stattdessen ist die Gesamtheit aller Gewichte der künstlichen Neuronen des NNs das Ergebnis des Trainings. Auch eine kleine Veränderung kann dazu führen, dass diese Gesamtheit verändert wird; ein neues, vollständig unbekanntes und nicht vorhersagbares Verhalten ist die Folge - man spricht vom „katastrophalen Vergessen“ (engl. *catastrophic forgetting*). Das System ist also nach dem Training nicht mehr lernfähig. In einem Roboter, der mit seiner Umwelt interagieren und von ihr lernen soll, ist ein solches System für die visuelle Verarbeitung daher häufig nicht sinnvoll.

Ein weiterer Grund, der gegen die Verwendung von ML-Systemen spricht, ist, dass sie teils unvorhersehbar darauf reagieren, wenn sich die Deployment-Umgebung von der Trainings-Umgebung unterscheidet, oder die Bewertungsfunktion für das Training nicht ausreichend präzise formuliert war, um unerwünschte Lösungswege auszuschließen.

Diese Problematik ist nach wie vor ein aktives Forschungsfeld, da es gerade für KI-Sicherheit von großer Bedeutung ist. Einige Beispiele aus dem Bereich der künstlichen Evolution für jene Sorte unerwartetes Verhalten, welche als *Reward Hacking* oder *Specification Gaming* bezeichnet wird, wurden von Lehmann et al. (2019) [10] in einer „verifizierten Sammlung von wissenschaftlich wichtigen und sogar unterhaltsamen Geschichten“ zusammengefasst (Übers. d. A.).

In einem Roboter, der nicht nur „unter Laborbedingungen“ agieren, sondern in direktem Kontakt mit Menschen stehen soll, sind Probleme dieser Art möglichst zu vermeiden. Aus diesen Gründen wurden Methoden des ML für diese Arbeit nicht weiter in Betracht gezogen.

2.3 Algorithmen zur Erkennung handschriftlicher Linien und Symbole

Viele Artikel, die sich mit der Erkennung von Strichzeichnungen beschäftigen, umgehen die bisher aufgeführten Komplexitäten der frühen Bildverarbeitung, indem sie eine andere Eingabemethode als eine Kamera verwenden. Dies wird bspw. durch die Nutzung elektronischer Tablets und Stifte erreicht.

Es muss betont werden, dass es sich bei dem Ergebnis einer Zeichnung auf einem Tablet nicht wirklich um ein (digitales) Bild im engeren Sinne handelt:

Ein digitales Bild kann als zweidimensionale Matrix verstanden werden, in der jede Zelle bzw. jedes Pixel einen Farbvektor beinhaltet, welcher die Farbe der Zelle (bspw. im RGB- oder YUV-Farbmodell) definiert.

Eine Zeichnung auf einem Tablet hingegen ist eine Liste von bereits zu *Strokes* (engl. für „Strich“) gruppierten x-y-t-Farb-Koordinaten. Dabei identifizieren die x- und y-Koordinaten die Position des Punktes auf der Zeichenebene und die t-Koordinate hält den Zeitpunkt fest, zu dem dieser Punkt gezeichnet wurde.

Die Farbwerte für jeden Punkt sind ebenfalls hinterlegt, doch die entscheidenden Unterschiede zu einem digitalen Bild sind zum einen die zusätzlichen Zeitkoordinaten, welche die zeitliche Abfolge festhalten, in der die am Stroke beteiligten Punkte gezeichnet wurden, und zum anderen die fehlende Notwendigkeit, die zu betrachtenden Bildelemente von einem Hintergrund zu trennen, da die Strokes bereits als Liste an Punkten vorliegen.

Ein Ansatz, welcher auf Strokes aufbaut, wird im Kontext der Zeichnungserkennung meist als „Online“ bezeichnet, während die Analyse eines Bildes als „Offline“ bezeichnet wird.

Cheng und Yang (1990) [11] entwickelten einen Offline-Algorithmus, welcher technische Zeichnungen automatisch erkennen sollte. Während sie nicht detailliert alle Schritte ihres Prozesses aufführen, scheint es dennoch offensichtlich, dass ihr Ansatz nicht gut generalisierbar sein dürfte: Nach der Vorverarbeitung der eingehenden Bilder ist der erste Schritt des sog. ARSAD Systems, die Seriennummern in der Zeichnung zu erkennen. Auch in den darauffolgenden Schritten ist das System basierend auf Vorwissen über die Struktur der erwarteten Bilder aufgebaut; auch wenn es für diese Anwendung also sehr gut funktionieren könnte, scheint es kein günstiger Ansatz für allgemeinere Erkennung zu sein.

Um die Techniken der Online-Ansätze auf Bilder anwenden zu können, wurden Algorithmen zur „*Stroke Extraction*“ entwickelt. Diese sollen aus Bilddaten Strokes generieren - soweit dies möglich ist.

Ein gutes Beispiel dafür liefert der Artikel von L'Homer (2000) [12]. In diesem Artikel werden aus einem unverrauschten Binärbild Schätzungen für die wahrscheinlich erfolgten Strokes extrahiert.

Ein wichtiger Fortschritt, den L'Homers Arbeit liefert, ist der Umgang mit sich kreuzenden Linien. Dies ist ein relevantes Problem, da ohne die zeitliche Dimension von Strokes unklar ist, ob bspw. der kleine Buchstabe „t“ aus einem, zwei, drei, oder vier Strokes bestehen soll. Vereinfacht gesagt wird mit dem vorgestellten Ansatz die Glätte der möglichen Kurven, die aus einer Kombination der an einer Kreuzung beteiligten Linien erzeugt werden könnten, verglichen, und die glatteste Kurve als die richtige angenommen.

Allerdings erfordert die vorangegangene Identifikation der Linien die Berechnung vieler Kreise, was die Methode *wahrscheinlich* davon abhält, eine gute Effizienz zu erreichen.[6]

Wróbel et al. (2019) [13] entwickelten den Ansatz von L'Homer weiter; in ihrem Artikel heißt es, die extrahierten Striche seien glatter, jedoch sei die Beschleunigung des Algorithmus ein Ziel zukünftiger Arbeit.

Ein gänzlich anderer Ansatz der Stroke Extraction wird von Bauer und Barrett (2016) [14] vorgestellt: Die Motivation für ihre Arbeit war es, die Schrift aus historischen Dokumenten zu digitalisieren; diese Dokumente sind häufig vergilbt oder anderweitig beschädigt, was die verlässliche Binarisierung erschwert. Der Dijkstra-Algorithmus, eine Methode zum Finden des günstigsten Weges zwischen Knoten eines Graphen, ist zentral für ihre Arbeit: Zunächst werden „Ausgangspunkte“ (engl. *seed points*) gefunden, indem nach dem Pfad der geringsten Helligkeit gesucht wird. Dazu werden an der oberen und unteren Kante des betrachteten Bildausschnitts äquidistant Start- bzw. End-Punkte des Dijkstra-Algorithmus gesetzt, wobei die Kosten eines jeden Pixels dem Helligkeitswert dieses Pixels entsprechen. Die Punkte, an denen sich mehrere Pfade überschneiden, werden als Ausgangspunkte verwendet und im nächsten Schritt werden diese Punkte iterativ miteinander verbunden, wieder unter Verwendung des Dijkstra-Algorithmus. Nach ein paar weiteren Schritten kann die Schreibbewegung approximiert werden, mit der das Dokument ursprünglich verfasst wurde.

[6] Genau lässt sich das auf Basis des Artikels nicht sagen, da weder Effizienzberechnungen, noch ein vollständiger Ablauf dargestellt werden.

Leider scheint es keine Berechnungen zur Effizienz dieser Algorithmen zu geben.

Diese Arbeiten zeigen, dass eine Approximation der ursprünglichen Stroke-Informationen aus (Offline-)Bildern prinzipiell möglich ist, was bedeutet, dass zumindest manche der Online-Ansätze auch für Offline-Problemstellungen Verwendung finden können, sofern sie nicht auf unwiederbringliche Informationen angewiesen sind.

Der für diese Arbeit als am wichtigsten eingeschätzte solche Ansatz ist der von Lee, Kara und Stahovich (2007) [15]. Die Autoren beschreiben in dieser Veröffentlichung einen lernfähigen Algorithmus zur Erkennung von Symbolen aus mehreren Strokes, welcher nicht die einzelnen Strokes als solche betrachtet, sondern die Beziehungen der einzelnen Strokes zueinander als Graph darstellt (ARG, „*attributed relational graph*“) und die Erkennung auf Basis dieses Graphen durchführt, was für hohe Robustheit sorgt. In einem ARG werden in den Knoten die Länge jeder Linie relativ zur Summe der Längen aller beteiligten Linien eingetragen, während in den Kanten für jedes Linienpaar die Menge der Schnittpunkte, deren Position entlang der Linien und der Schnittwinkel festgehalten werden; auch die Knoten von Linien, die sich nicht schneiden, haben eine gemeinsame Kante.

Soll ein Symbol erkannt werden, wird die Unähnlichkeit zu allen bekannten Symbolen mittels sechs verschiedener Fehlermaße errechnet: Der Unterschied in der Anzahl der Linien, Abweichungen der vorliegenden Linientypen - unterschieden werden gerade und gekrümmte Linien - Längenabweichungen sowie Differenzen in der Anzahl, dem Winkel und der Position der Schnittpunkte beteiligter Linien. Diese Fehlermaße gehen dann mit empirisch ermittelten Gewichten in die abschließende Ähnlichkeitsbewertung ein.

Um ein neues Symbol zu erlernen, wird aus mehreren Beispielen ein durchschnittlicher ARG erzeugt. Die Menge der nötigen Beispiele ist dabei relativ gering: Von einem Experiment wird für die kombinierte Ähnlichkeitsbewertung mit nur sechs Beispielen eine „*top-one accuracy*“ von über 95 % und eine „*top-three accuracy*“ nahe 98 % berichtet. Während die Erkennung von Symbolen rotationsinvariant ist, muss beim Erlernen entweder eine über alle Beispiele konsistente Orientierung des Symbols vorliegen oder immer die gleiche Reihenfolge beim Zeichnen der einzelnen Strokes verwendet werden - im Kontext dieser Arbeit müssen die zu erlernenden Symbole also immer gleich ausgerichtet sein, da die zeitliche Komponente von Strokes nicht verfügbar ist.

Bevor es zu einer Erkennung spezifischer Symbole kommen kann, müssen die Symbole im Bild lokalisiert werden.

Bhattacharya et al. (2020) [16] beschreiben einen Offline-Ansatz zur Lösung dieses Problems am Beispiel eines handgezeichneten elektrischen Schaltplans. Nach etwas Vorverarbeitung wenden sie die morphologische Schließen-Operation mit einem runden Strukturelement auf das Bild an. Dadurch werden die Bildregionen, welche ein Symbol beinhalten, zu sog. *Blobs*, während die Kabelverbindungen größtenteils unverändert bleiben. Nachdem im nächsten Schritt die Kabelverbindungen mittels zweier invertierter Anwendungen des „*run length smoothing algorithm*“ (RLSA) entfernt werden, bleiben die *Blobs* zurück, und es können Bounding Boxes um sie gezeichnet werden.

Die Größe des gewählten Strukturelements relativ zu der Größe eines gegebenen Symbols kann dabei entscheidend dafür sein, ob dies Symbol zu einem *Blob* wird oder größtenteils unverändert bleibt. Dies hat zur Folge, dass nicht nur einzelne Linien, sondern auch Symbole, die größer als erwartet sind, mit dieser Technik aus dem Bild entfernt werden; es ist also ein Mindestmaß an Vorwissen über die erwartete Symbolgröße von Nöten.

3 Entwicklung des Algorithmus

Die Struktur dieses Kapitels wurde inspiriert von Dörner [17]. In seinem Buch „Bauplan für eine Seele“ führt Dörner die Lesenden durch ein Gedankenexperiment, in dem durch die schrittweise Erweiterung eines zu Beginn einfachen Systems sehr komplexes Verhalten entstehen kann. Zu diesem Zweck bringt Dörner zunehmend mehr Anforderungen an das System zur Sprache, welche es bewältigen können muss, um relativ autonom existieren zu können, und löst diese dann durch schrittweise Erweiterung.

Davon inspiriert wird in diesem Abschnitt die Komplexität der Aufgabe, Strichzeichnungen in der echten Welt erkennen zu können, Schritt für Schritt herausgearbeitet: Es wird zunächst eine einfache Ausgangssituation angenommen, in der das Ziel eines lernfähigen Systems zur Strichzeichnungserkennung auf Basis des aktuellen Forschungsstands vergleichsweise leicht umsetzbar wäre (vgl. Abs. 2.3). Anschließend wird die Ausgangssituation schrittweise an die in dieser Arbeit vorliegende Situation angepasst, während das Ziel unverändert bleibt; dabei werden die entstehenden Probleme und mögliche Lösungen aufgezeigt. Dieser Abschnitt dient so als Kontext für die in Abs. 3.1 vorgestellten Schritte.

Als Ausgangssituation dieses Abschnitts wird angenommen, dass die gesuchten Linien bereits digital als Strokes (vgl. Abs. 2.3) vorliegen. In diesem Fall ist die Verwendung eines der in Abs. 2.3 vorgestellten Online-Algorithmen ausreichend.

Sobald die Informationen nicht mehr als Strokes, sondern als Bild vorliegen, wird die Aufgabe schwieriger. Der einfachste solche Fall ist ein unverrauschtes Binärbild mit perfekter Trennung zwischen den gesuchten Linien und dem Hintergrund. Um auch mit diesem Bild arbeiten zu können, kann das betrachtete System um eine der zuvor vorgestellten Stroke Extraction-Methoden erweitert werden.

Graustufen

Liegt das Bild nicht binär, sondern in Graustufen vor, so ist zunächst eine Trennung von Linien und Hintergrund erforderlich. Unter günstigen Bedingungen, wie einem guten Dokumentenscan, könnte als Verfahren bspw. die von Otsu (1979) [18] vorgestellte und inzwischen als „Otsu’s Method“ bekannte Methode angewandt werden, um die Linien vom Hintergrund zu trennen.

Bei diesem Verfahren wird ein optimaler „Schwellwert“ (engl. *Threshold*) ermittelt, mit dem Helligkeitswerte von Pixeln zu zwei oder mehr Gruppen zusammengefasst werden können. Das Verfahren arbeitet allerdings meist „global“, was bedeutet, dass für das gesamte Bild ein einziger Schwellwert berechnet wird; dies wird zum Problem, wenn die Helligkeitsverteilung global nicht klar trennbar ist, was bspw. bei einem im Bild vorliegenden Helligkeitsgradienten häufig der Fall ist. Liegt ein solcher Helligkeitsgradient vor, so kann dies mit Methoden, bei denen ein lokaler Schwellwert berechnet wird, häufig ausgeglichen werden. Ein Beispiel für eine solche Methode wäre die von Niblack (1986, zitiert nach Sauvola und Pietikäinen, 2000, [19]) oder ihre Erweiterung, die von Sauvola und Pietikäinen (2000) [19] entwickelt wurde. Bei zu harten Schattenkanten können jedoch auch diese Methoden an Effektivität verlieren.

Rauschen

Bislang wurde ein unverrauschtes Bild angenommen, doch in realistischeren Bildern ist Rauschen unvermeidbar. Das Rauschen in einem Bild kann als eine Art „optisches Störgeräusch“ verstanden werden. Solches Rauschen kann durch verschiedene *externe* Ursachen auftreten, wie zum Beispiel durch ein hochfrequentes Flackern der Lichtquelle, durch aus der Oberflächenstruktur des Hintergrunds entstehende kleine Schatten oder durch Staub und vergleichbare kleine Verunreinigungen, und gleichermaßen auch durch *interne* Ursachen wie Sensorrauschen der Kamera, Limitierungen der Aufnahmefähigkeit (bspw. durch fehlende Bildtiefe bei starken Helligkeitskontrasten/Dunkelheit) oder automatische Bildkorrekturen.

Zum Entfernen von Rauschen gibt es viele etablierte Methoden, wie die Verwendung von Gauß-Filtern oder morphologischen Operationen (vgl. App. A.3), welche in einem Tiefpassfilter resultieren. Diese Methoden sind meist sehr effektiv, allerdings resultiert eine Filteroperation immer in Informationsverlust.

Während dies natürlich bei Menschen ebenfalls der Fall ist (vgl. Abs. 1.2.5), nutzt der entsprechende Prozess im menschlichen Sehen kein fest eingestelltes Filter - stattdessen wird stark abstrahiert und auf diese Weise eine Datenreduktion und Rauschunterdrückung realisiert. Dies ist flexibler als ein fest eingestelltes Filter:

Stehen wir vor einer schwarzen Linie auf einer weißen Wand, so können wir bewusst unsere Aufmerksamkeit lenken. Es ist uns möglich, nur die Linie als solche wahrzunehmen, die Lichtreflexionen auf der schwarzen (oder weißen) Farbe, die leicht unsauberen Kanten der schwarzen Linie, Staub, welcher an der Wand haftet, die Oberflächenstruktur der Wand - und wie sie sich zwischen dem schwarzen und weißen Bereich unterscheidet -, etc. Wir haben also die Möglichkeit, bewusst Kontrolle darauf auszuüben, welche Details wahrgenommen und welche ausgeblendet werden. Würde das Bild in der menschlichen Wahrnehmung jedoch vorab immer so gefiltert, dass die schwarze Linie vom weißen Hintergrund isoliert wird und als „ein Objekt“ vorliegt, so wäre ein Wahrnehmen dieser Details nicht mehr möglich.

Hinzu kommt, dass das gleiche Bildelement kontextabhängig relevant oder irrelevant sein kann - so ist das Gitter auf einem karierten Blatt Papier meist nicht wichtig, sobald darauf gezeichnet wurde, doch auf einem Go-Brett ist das Gitter ein wichtiger Bestandteil des Spiels, unabhängig davon, wie viele Spielsteine im Spiel sind. Das Problem, welches als nächstes zu lösen ist, ist also, dass der Filterprozess des betrachteten Systems kontextabhängig sein sollte.

Kontext

Das System müsste also in der Lage sein, manche Bildelemente über andere zu priorisieren, ohne die anderen *immer* aus dem Bild zu entfernen, da es sonst für diese blind würde. Es kann Inspiration aus der menschlichen visuellen Verarbeitung gezogen werden: Wie in Abs. 2.1 dargestellt wurde, gibt es bereits Methoden zur Erzeugung einer „Wichtigkeits-Karte“.

Unter Verwendung einer solchen Wichtigkeits-Karte könnte nun immer der als am wichtigsten bewertete Bildanteil betrachtet werden und nach und nach das gesamte Bild erfasst werden. Im Kontext des ursprünglichen Ziels der Erkennung von Strichzeichnungen, könnten auf Basis der Wichtigkeits-Karte Linien erzeugt werden, ähnlich dazu, wie es mit einem zuvor betrachteten Graustufenbild möglich wäre.

Weiter wird dieses Gedankenexperiment an dieser Stelle nicht durchgeführt, da hier der aktuelle Arbeitsstand erreicht ist, welcher nun vorgestellt wird.

3.1 Präsentation des entwickelten Algorithmus zur Extraktion von Linien

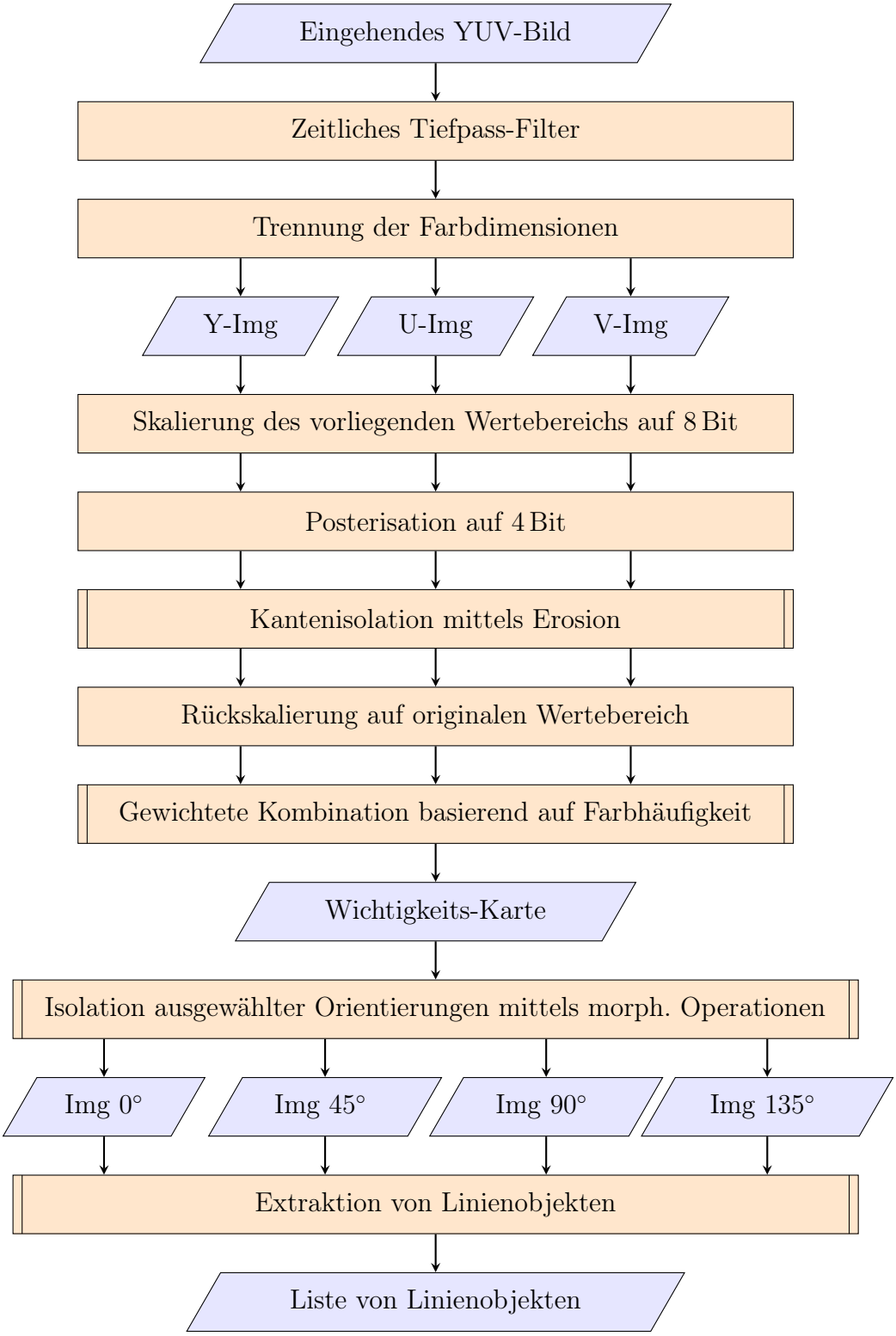
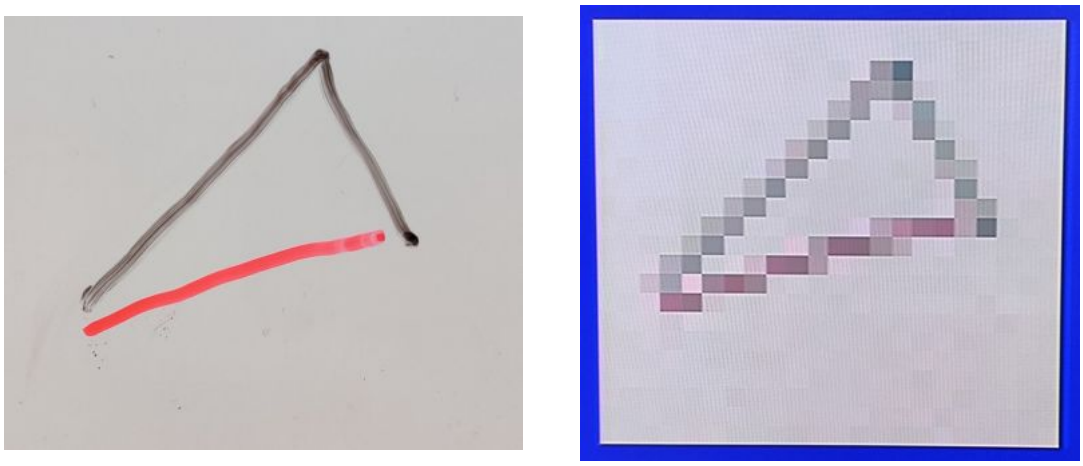


Abbildung 6: Datenflussdiagramm des entwickelten Algorithmus zur Linienextraktion, welcher in diesem Abschnitt vorgestellt wird. Als Verständnishilfe ist der Schritt „Extraktion von Linienobjekten“ im Appendix in Abb. 34 zusätzlich graphisch dargestellt. Ebenfalls vorliegend ist eine Variante *dieser* Abbildung (Abb. 6) mit Bildern der Zwischenergebnisse - siehe dazu Abb. 33.

Im Folgenden wird der in dieser Arbeit entwickelte und in Abb. 6 dargestellte Algorithmus erläutert. Das Ziel ist dabei, die im Bild vorhandenen Linien zu erkennen. Zu diesem Zweck wird zunächst eine Wichtigkeits-Karte (vgl. Abs. 1.2.5) generiert.

Als Ausgangspunkt des in dieser Arbeit entwickelten Algorithmus (schematisch in Abb. 6 und in App. B dargestellt) wird ein Bild nach der Vorverarbeitung mittels PixelPipeline (vgl. Abs. 1.1.1) angenommen: Das Ausgangsbild ist 22×22 Superpixel groß (vgl. Abs. 1.1.2), die Farbwerte je Superpixel liegen als 8 Bit YUV-Werte vor (Abs. 1.1.3) und der Farbwert jedes Superpixels ist der Durchschnitt der Farbwerte aller beteiligten Pixel. Ein solches Ausgangsbild ist in Abb. 7 dargestellt.



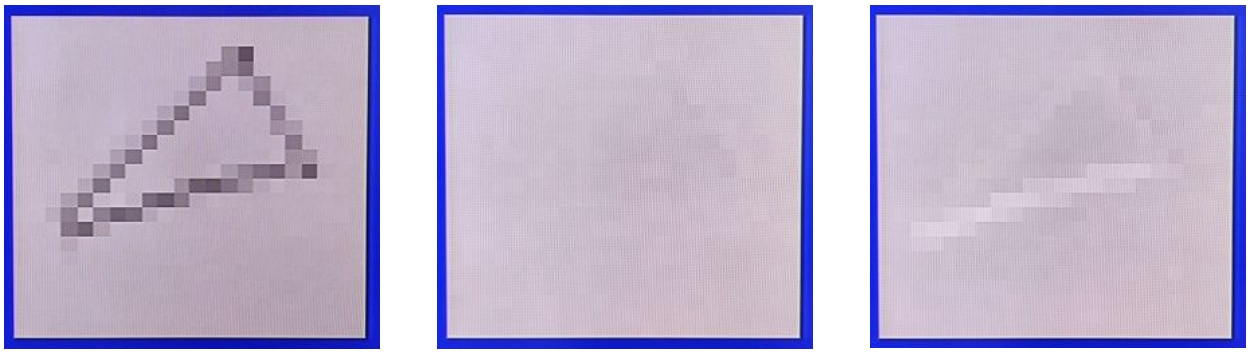
(a) Kamera Input

(b) Ausgabe der PixelPipeline

Abbildung 7: Arbeitsbeispiel für den Algorithmus ist das in beiden Bildern dargestellte Dreieck - das linke Bild wurde direkt mit einer Handykamera aufgenommen, während das rechte das Ergebnis von Myons Bildvorverarbeitung durch die PixelPipeline ist. Es ist anzumerken, dass der rote Strich im linken Bild sehr viel markanter scheint als im rechten Bild. Dies liegt zum Teil an der Bildung der Superpixel: Sowohl der rote Strich als auch der Hintergrund hatten auf die „roten“ Superpixel Einfluss. Die andere Ursache liegt darin, dass Kameras Farben anders verarbeiten als Menschen; in App. A.2 wird dies näher erläutert.

Das eingehende Bild wird mit einem zeitlichen Tiefpassfilter gefiltert, um von der Kamera ausgehendes Sensorrauschen zu reduzieren. Anschließend wird das Bild nach Farbdimensionen getrennt. Es liegen nun drei Bilder vor - dargestellt in Abb. 8 -, von denen jedes nur eine der drei Farbdimensionen, jeweils mit einem maximal möglichen Wertebereich von $[0, 255]$ (8 Bit), beinhaltet.

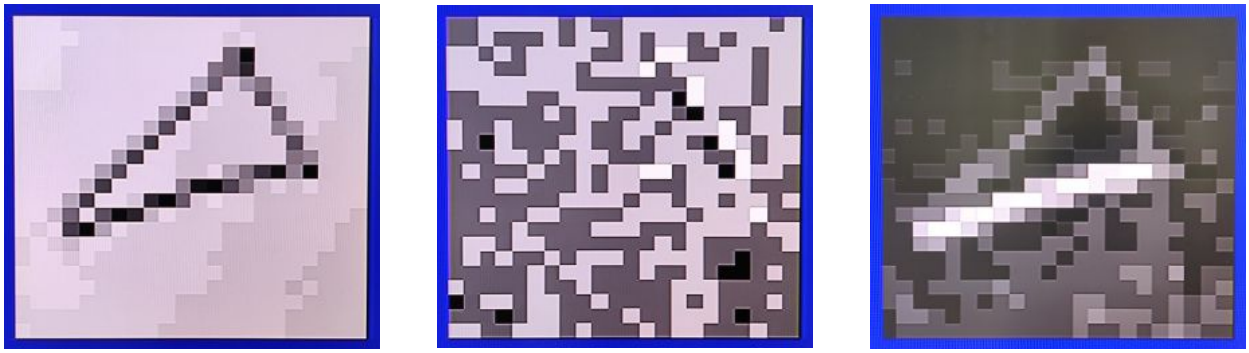
Für jedes Bild wird ein Skalierungsfaktor berechnet und angewandt, welcher so gewählt ist, dass der 8 Bit Wertebereich im Anschluss vollständig ausgefüllt wird. Dies ist nötig, da im nächsten Schritt die Farbtiefe reduziert wird und ohne diese Trennung und Skalierung markante Details, die für Menschen auffällig wären, verloren gehen könnten (vgl. App. A.2). Ein Beispiel für ein solches Detail ist in Abb. 7 und Abb. 8 zu erkennen: Der im Kamera Input (Abb. 7a) klar als Rot erkennbare Strich scheint nach der Verarbeitung durch die PixelPipeline nur noch schwach rötlich (Abb. 7b). Dies fällt nach der Trennung der Farbkanäle noch stärker auf; der *rote* Strich ist auf dem Y-Kanal (Abb. 8a), welcher Helligkeit anzeigt, klarer zu sehen als auf dem V-Kanal (Abb. 8c), auf dem eine klarere Abhebung vom umliegenden Grau zu erwarten gewesen wäre.



(a) Y-Kanal (b) U-Kanal (c) V-Kanal

Abbildung 8: Darstellung der einzelnen Farbkanäle des Bildes aus Abb. 7. Die einzelnen Kanäle werden für erhöhte Differenzierbarkeit in Graustufen dargestellt. Eine kurze Einführung des YUV-Farbraums ist in Abs. 1.1.3 gegeben.

Anschließend an die Skalierung wird also die Farbtiefe (der mögliche Wertebereich) der Bilder von 8 Bit auf 4 Bit reduziert, was als „Posterisation“ bezeichnet wird. Danach wird die Häufigkeit jedes der verbleibenden 16 Farbwerte pro Bild gezählt. Es liegen nun also drei Bilder mit einer Farbtiefe von 4 Bit vor:

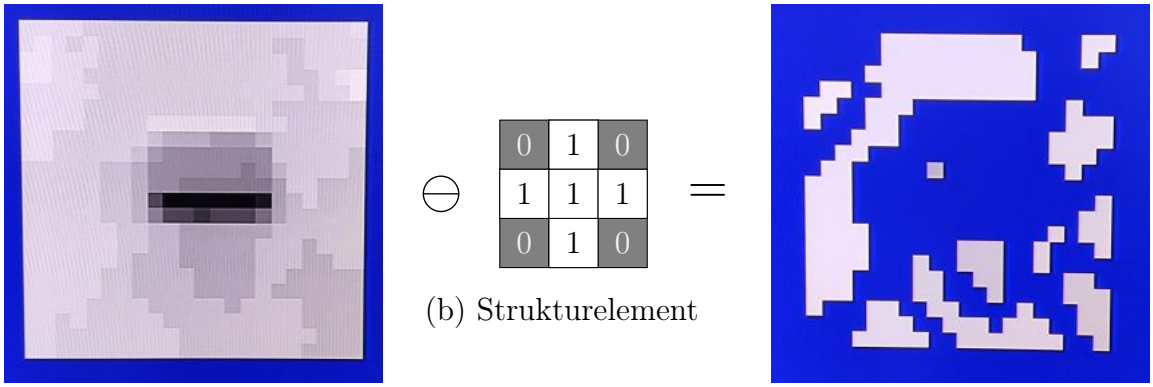


(a) Y-Kanal (b) U-Kanal (c) V-Kanal

Abbildung 9: Ergebnis der Skalierung und Posterisation der Bilder aus Abb. 8. Da auf dem U-Kanal im vorherigen Bild (Abb. 8b) nur sehr kleine Veränderungen vorlagen, wurde durch diese Operationen auch das Rauschen im Bild verstärkt. Gleiches gilt für das Bild des V-Kanals in Abb. 8c, wenn auch in geringerem Maße, da dort der rote Strich ein markantes Detail darstellt, welches dominanter ist als das Rauschen.

Für die Aufgabenstellung der Arbeit sind nur Striche relevant, weshalb nun die Kanten farblich homogener Bildbereiche isoliert werden; eine solche Kante besitzt eine Dicke von einem Pixel, somit bleiben ausreichend schmale (farblich homogene) Bildbereiche unverändert, während von breiteren Bildbereichen nur die Kanten verbleiben. Die zuvor durchgeführte Posterisation ist für diesen Schritt essenziell, da durch die reduzierte Farbtiefe leichter solche farblich homogenen Bildbereiche entstehen.

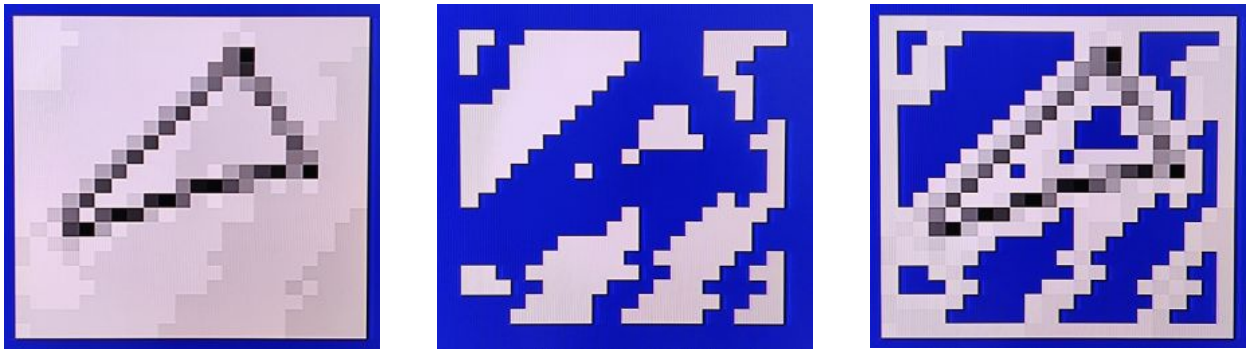
Um diese Kantenisolation zu realisieren, wird zunächst eine Variante der Erosion auf das Bild angewandt (für eine Beschreibung der herkömmlichen Erosion siehe App. A.3). Üblicherweise wird die Erosion auf ein binäres Bild angewandt, in diesem Fall liegt jedoch ein Bild mit 16 Farbwerten vor, weshalb nicht binär auf „1“ oder „0“ geprüft wird, sondern auf Gleichheit: *Alle Zellen*, welche in dem verwendeten Strukturelement den Wert „1“ haben, müssen in dem zu bearbeitenden Bild den *gleichen* Farbwert besitzen. Ist dies der Fall, so wird *dieser* Farbwert in das resultierende Bild eingetragen, andernfalls wird an die entsprechende Stelle der Wert „0“ gesetzt. Ein Beispiel dieser Variation wird in Abb. 10 dargestellt.



(a) Bild vor Erosion (b) Strukturelement (c) Ergebnis der Erosion

Abbildung 10: Beispiel für Erosionsvariante mit Gleichheit. In Abb. 10c verbleiben nur diejenigen Pixel der farblich homogenen Bildbereiche, welche keine Kante darstellen - alle Bereichsübergänge wurden entfernt (blaue Bereiche im Bild). Eine Anwendung auf das Arbeitsbeispiel ist in Abb. 11 gegeben.

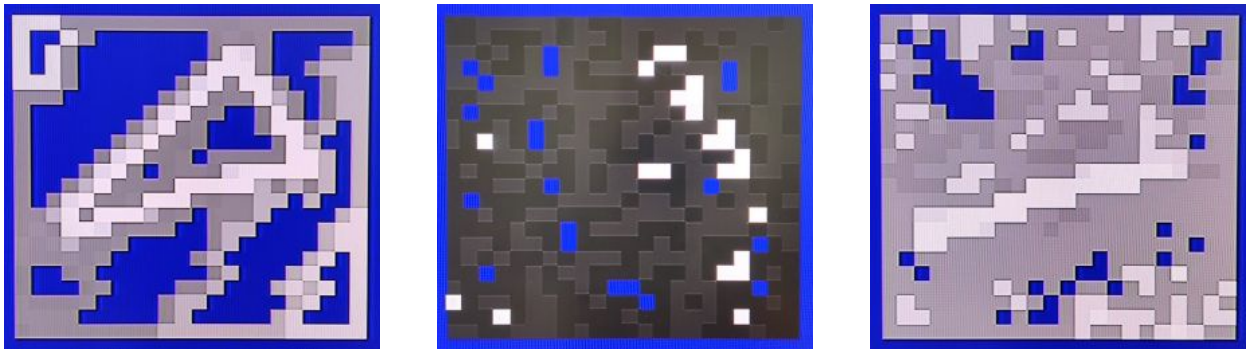
Das Ergebnis ist ein Bild, in dem die äußersten Pixel jedes farblich homogenen Bildbereiches entfernt wurden. Dies wird im Anschluss von dem Bild, das vor Anwendung der Erosion vorlag, abgezogen: $\mathbf{I}' = \mathbf{I} - (\mathbf{I} \ominus \mathbf{S})$. Es verbleiben nur die Kanten im Bild (siehe Abb. 11).



(a) Y-Kanal vor Erosion (b) Y-Kanal nach Erosion (c) Isolierte Kanten

Abbildung 11: Prozess der Kantenisolation durch Erosion am Beispiel des Y-Kanals. Die Bereiche des Bildes, die in Abb. 11c erhalten blieben, sind die Kanten der gleichfarbigen Bildbereiche in Abb. 11a. Die entfernten Bereiche werden nicht als Graustufenwert, sondern in Blau gezeigt, um die Unterscheidung zu dunklen Bereichen klarer zu gestalten. Analog wird mit den U- und V-Kanälen verfahren.

Als Grundlage des nächsten Schritts wird angenommen, dass die seltenen Farbwerte diejenigen sind, welche von Interesse sind, da es in dieser Arbeit um Linienerkennung geht und einzelne Linien zumeist nur einen kleinen Anteil des Bildes ausmachen - im Arbeitsbeispiel wären Rot- und Schwarztöne seltene Farben, während Weißtöne häufig sind. Auf Basis dieser Annahme werden nun die Farbwerte im Bild so verändert, dass die bislang häufigsten Farbwerte mit niedrigen Werten und die bislang seltensten Farbwerte entsprechend mit hohen Werten *ersetzt* werden. Es handelt sich ab jetzt also um Falschfarbenbilder. Diese Kantenisolation und folgende Umgewichtung soll zusammengenommen in einer global angewandten Approximation dessen resultieren, was im V1 durch die Nachbarschaft von Neuronen und deren Interaktion erreicht wird (vgl. Abs. 1.2.5). Das Ergebnis ist in Abb. 12 gegeben:



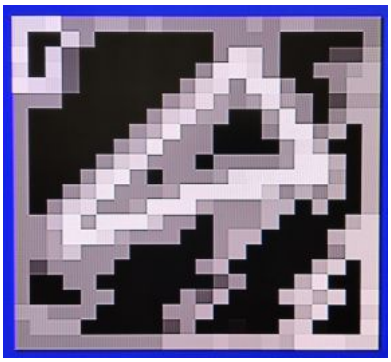
(a) Y-Kanal (b) U-Kanal (c) V-Kanal

Abbildung 12: Aus der Umgewichtung resultierende Falschfarbenbilder der zuvor isolierten Bildkanten. Die Helligkeit sagt nun etwas über die Häufigkeit der ursprünglichen Farbe des Pixels aus: Je heller der jetzige Farbwert, desto seltener kam die Farbe im Bild vor - bspw. ist das zuvor schwarze Dreieck in Abb. 12a nun mit Weiß dargestellt. Bereiche, die nicht mehr zum Bild gehören, wurden blau hinterlegt (vgl. Abb. 11).

Um die gewünschte Wichtigkeits-Karte zu erhalten, werden die drei Bilder wieder zusammengeführt. Dazu werden sie zunächst wieder auf den ursprünglichen Wertebereich zurückskaliert, dann wird pixelweise der Durchschnitt der drei Bilder gebildet. Anschließend wird das gesamte Bild wieder auf den vollen Wertebereich von 8 Bit skaliert und zurück auf 4 Bit posterisiert.

Diese abschließende Skalierung und Posterisation erfolgt, um einen einheitlichen „Schwellwert“ (engl. *Threshold*) definieren zu können; sie garantiert, dass es immer Punkte im Bild geben wird, welche oberhalb des Schwellwert liegen. Die semantische Bedeutung des Schwellwerts ist, dass die ungewöhnlichsten (im Folgenden „interessantesten“) Bildbereiche isoliert betrachtet werden. Der hier verwendete Schwellwert ist „13“, es werden also nur Pixel mit den obersten drei (Falsch-)Farbwerten weiter betrachtet. Dieser Schwellwert wurde empirisch ermittelt, kann jedoch bei Bedarf angepasst werden. Als Ergebnis der skalierten und posterisierten Zusammenführung der drei Farbkanäle liegt nun eine sehr rudimentäre Wichtigkeits-Karte vor, auf welche der Schwellwert angewandt wird (Abb. 13a).

Um aus den Kanten in der Wichtigkeits-Karte nun Linien zu bilden, wird angenommen, dass nach Anwendung des Schwellwerts nur noch die Kanten der interessantesten Bildbereiche vorliegen. Die Anwendung des Schwellwerts resultiert in einem binären Bild (vgl. Abb. 13b), aus dem nun eine Liste von Linien erzeugt werden kann.



(a) Wichtigkeits-Karte vor Anwendung des Schwellwerts

(b) Wichtigkeits-Karte nach Anwendung des Schwellwerts

Abbildung 13: Darstellung der Wichtigkeits-Karte vor und nach der Anwendung des Schwellwerts. Der Schwellwert ist so gewählt, dass die drei höchsten Werte erhalten bleiben, was sich experimentell als am günstigsten herausgestellt hat.

Während in Abs. 2.3 verschiedene Methoden der Stroke Extraction dargestellt wurden, arbeiteten diese auf Bildern höherer Auflösung. Es wurde mit dem Ziel einer einfacheren Implementierung daher eine simplere Methode der Stroke Extraction entwickelt, die auf Basis morphologischer Operationen funktioniert:

Die Wichtigkeits-Karte wird mit vier 3×3 Strukturelementen morphologisch geöffnet (für morphologisches Öffnen siehe App. A.3). Jedes SE stellt eine Linie dar, welche eine Orientierung von 0°, 45°, 90° oder 135° besitzt. Die vier resultierenden „Orientierungsbilder“ beinhalten also nur noch entlang der jeweiligen Orientierung zusammenhängende Bildbereiche (siehe Abb. 14).

Im nächsten Schritt wird aus den Orientierungsbildern eine Liste von „Linienobjekten“ erzeugt. Es wird der Begriff „Linienobjekt“ verwendet, um den Unterschied zu den in Abs. 2.3 erwähnten „Strokes“ zu betonen: Während Strokes aus vielen Punkten bestehen, welche nicht nur ihre Position, sondern auch ihre zeitliche Abfolge codieren, handelt es sich bei einem Linienobjekt nur um zwei Punkte in der x-y-Ebene - den Ausgangspunkt und den Endpunkt der Linie - und die Information, aus welchem der Orientierungsbilder es generiert wurde.

Um aus den vorliegenden Orientierungsbildern nun eine Liste an Linienobjekten zu erhalten, wird rekursiv vorgegangen: Das erste Pixel mit dem Wert „1“ (im Folgenden „vorhandene“/„existierende“ Pixel) in einem der Orientierungsbilder wird als Ausgangspunkt gewählt und ein Linienobjekt mit diesem Pixel als Ausgangs- und Endpunkt sowie der relevanten Orientierung initialisiert.

Dieser Schritt ist semantisch ähnlich zu den von Bauer und Barrett verwendeten *seed points* (vgl. Abs. 2.3), allerdings ist das Finden der hier verwendeten Ausgangspunkte um einiges leichter, da durch die Anwendung des Schwellwerts nur noch in Frage kommende Pixel im Bild vorkommen.

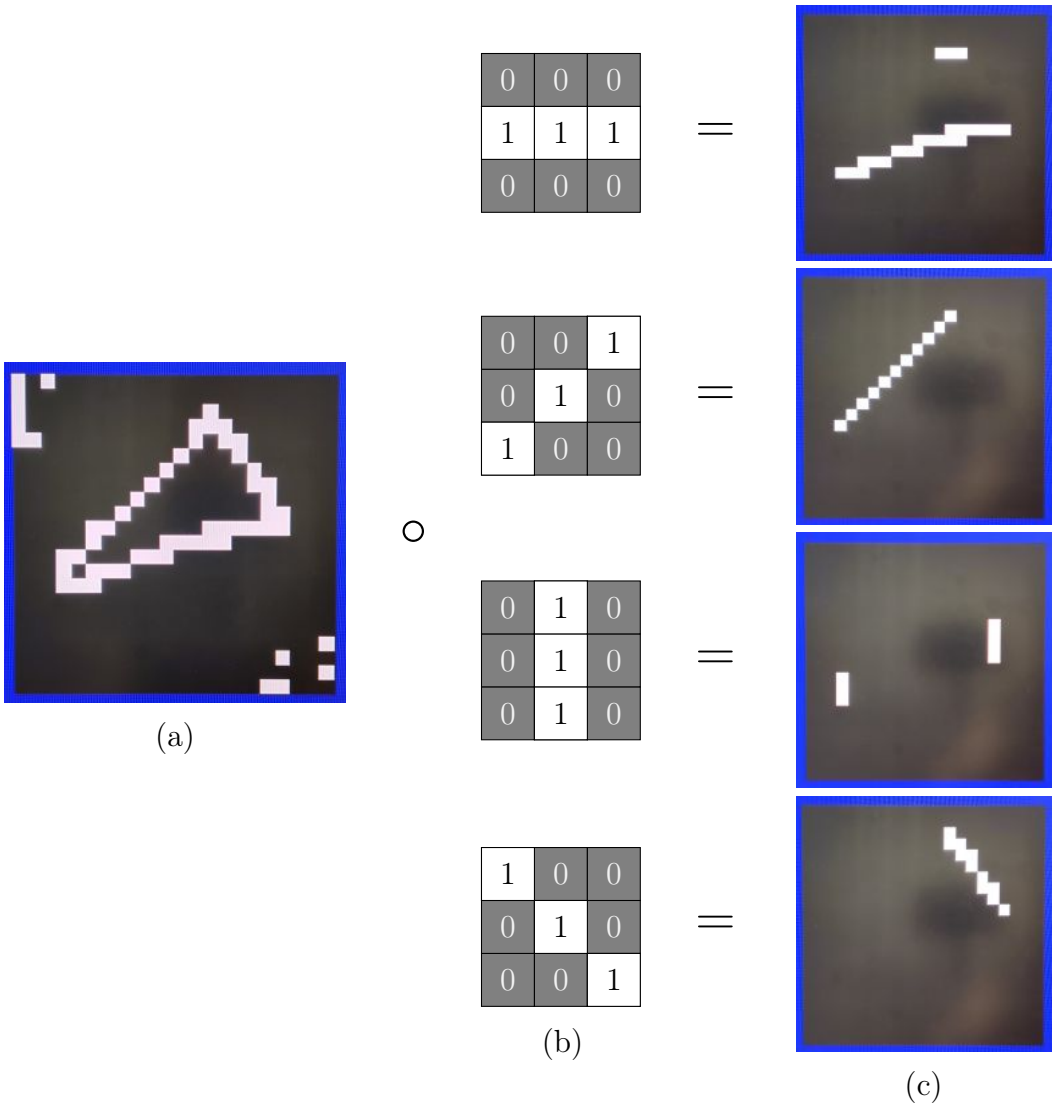


Abbildung 14: Isolation der Orientierungen mittels morphologischem Öffnen.
(a): Binäre Wichtigkeits-Karte, (b): Strukturelemente für vier Orientierungen,
(c): Ergebnis der Öffnung: $a \circ b_i = c_i$.

Anschließend an die Initialisierung wird entlang der Orientierung dieses Orientierungsbildes (der relevanten Orientierungsachse) nach benachbarten Pixeln gesucht.[7] Dabei werden die Pixel bevorzugt, welche direkt auf der relevanten Orientierungsachse liegen, doch auch die gemeinsamen Nachbarn mit diesen werden betrachtet - eine Visualisierung ist im Appendix gegeben, siehe Abb. 35. Existiert ein solches Pixel, wird der Ausgangs- oder Endpunkt des Linienobjekts auf dieses Pixel verlegt, und die Nachbarn desjenigen Pixels, welches nun nicht mehr der Ausgangs- oder Endpunkt des Linienobjekts ist, werden aus dem Orientierungsbild entfernt.[8]

[7]Mit „benachbarten“ Pixeln sind jene mit einer Chebyshev-Distanz von „1“ zum betrachteten Pixel gemeint. Für eine umfassendere Erläuterung des Begriffs siehe App. A.4.

[8]Ob es sich um den Anfang oder das Ende der Linie handelt ist nur für die Implementierung relevant.

Dies erfolgt rekursiv, bis das Linienobjekt nicht weiter wachsen kann, weil keine Nachbarn mehr existieren. Sobald keine Nachbarn mehr gefunden werden, wird geprüft, ob Ausgangs- und Endpunkt des Linienobjekts identisch sind; ist dies der Fall, wird das Linienobjekt verworfen. Der rekursive Prozess der Bildung von Linienobjekten ist als Verständnishilfe im Appendix in Form eines Programmablaufplans visuell dargestellt, siehe Abb. 34.

Nachdem das Linienobjekt also abschließend bearbeitet wurde, wird nach einem Ausgangspunkt für das nächste Linienobjekt gesucht. Hier ist keine Rekursion nötig, da jedes Pixel pro Orientierung nur einem Linienobjekt zugehörig sein kann - mit Erreichen des Endes des Orientierungsbildes ist also sichergestellt, dass alle Linienobjekte dieser Orientierung gefunden wurden. Die verbleibenden Orientierungsbilder werden auf die gleiche Weise bearbeitet.

Das Ergebnis ist eine Liste von Linienobjekten, die nun als Basis für eine anschließende Symbolerkennung genutzt werden kann. Die erkannten Linien sind in Abb. 15 eingezeichnet. Zu beachten ist, dass die resultierenden Linienobjekte nicht nur in den vier Orientierungen vorkommen, mit denen zuvor geöffnet wurde, was daran liegt, dass - wie zuvor erwähnt - bei der Linienbildung auch benachbarte Pixel betrachtet wurden: In einer Situation, in der ohne die Betrachtung von benachbarten Pixeln zwei benachbarte parallele Linienobjekte entstanden wären, wird stattdessen ein Linienobjekt gebildet; dieses kann ggf. den Ausgangspunkt des einen hypothetischen Linienobjekts und den Endpunkt des anderen besitzen. Das Linienobjekt hätte in diesem Fall also eine neue Orientierung, die keiner der zur Öffnung verwendeten Orientierungen entspricht. Ein Beispiel für ein solches Linienobjekt ist die untere Kante des in Abb. 15 dargestellten Dreiecks, deren Orientierung zwischen 0° und 45° liegt. Dieses Linienobjekt ist das Amalgam mehrerer hypothetischer Linienobjekte mit einer Orientierung von 0°, welche in der Wichtigkeits-Karte im Hintergrund noch zu sehen sind.

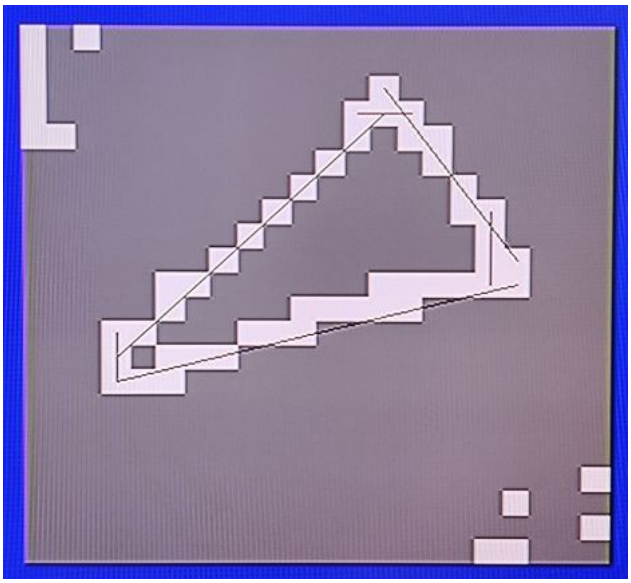


Abbildung 15: Darstellung der gefundenen Linienobjekte. Die Linienobjekte sind über die Wichtigkeits-Karte gezeichnet worden, in der die über dem Schwellwert liegenden Superpixel weiß hinterlegt wurden. In diesem Bild wurden sechs Linienobjekte erkannt.

3.2 Perspektiven für die Weiterentwicklung

In Abb. 15 ist zu erkennen, dass nicht nur die „gewünschten“ drei Linienobjekte (im Folgenden nur noch als „Linien“ bezeichnet) erkannt wurden.

Aufgrund von Zeitmangel wurde weder ein System zum Umgang mit den „unerwünschten“ Linien noch die Realisierung einer Implementierung des Algorithmus von Lee et al. (vgl. Abs. 2.3) auf Basis dieser Linien erreicht.

Eine Handhabung der unerwünschten Linien könnte wie folgt aussehen:

Zunächst werden die längsten erkannten Linien identifiziert; dazu könnten z.B. alle Längen berechnet und der Median der verschiedenen Längen als untere Grenze verwendet werden. Die Linien mit einer Länge über dieser Untergrenze könnten dann als „erste Näherung“ die Grundlage für die Symbolerkennung bieten. Ist das Ergebnis nicht eindeutig, so könnten dann weitere Schritte erfolgen: Kurze Linien, die eine längere Linie mit ähnlicher Orientierung schneiden, könnten mit dieser kombiniert werden. Diese Kombination könnte wie folgt aussehen: Es werden die beiden Endpunkte ausgewählt, die entlang der gemittelten Orientierung beider Linien am weitesten voneinander entfernt liegen. Aus diesen wird eine neue Linie gebildet, während die beiden ursprünglichen Linien verworfen werden.

Alternativ könnte das Ergebnis des vorherigen Schrittes - der Symbolerkennung auf Basis der ersten Näherung - verwendet werden: Die Symbolerkennung würde diejenigen bereits bekannten Symbole identifizieren, die der ersten Näherung am ähnlichsten sind. Die Differenz zwischen diesen Symbolen und der ersten Näherung könnte dann mit den kürzeren Linien verglichen werden, um zu testen, ob manche der kurzen Linien die Ähnlichkeit zu einem der Symbole erhöhen würden und so Klarheit geschaffen werden kann.

In beiden Fällen wäre es auch möglich, die für die Entscheidung kritischen Bildbereiche im nächsten Bild vergrößert zu betrachten.

Ebenfalls konzeptionell betrachtet, jedoch nicht vollends realisiert, wurde ein System zur gezielten Verschiebung des betrachteten Bildbereichs in Reaktion auf die Auswertung des Gesehenen. Dies war eine zentrale Motivation dafür, die Wichtigkeits-Karte zu implementieren:

Meist werden Symbolerkennung und Stroke Extraction als Werkzeuge verwendet, die gezielt auf Zeichnungen angewandt werden: Das System muss zwar erkennen, *was* dargestellt ist, jedoch nicht, ob das Bild verändert werden muss, um ein Symbol erkennen zu können. Diese Arbeit wird von den Menschen gemacht, welche die Symbolerkennung verwenden.

Im Falle dieser Arbeit ist der Anwender jedoch Myon, welcher bestenfalls nicht erst dazu angewiesen werden muss, ein Symbol zu erkennen.

Statt auf Anweisung zu warten, könnte der Algorithmus zur Linien- und Symbolerkennung aktiv werden, sobald die visuelle Verarbeitung nicht anderweitig benötigt wird. Myon könnte dann das aktuelle Blickfeld nach Symbolen absuchen und als wichtig erkannte Bildbereiche vergrößert betrachten.

Auf diese Weise könnte die Wichtigkeits-Karte eine Möglichkeit für Verhalten bieten, welches menschlichem Verhalten näher kommt, da auf Grundlage der Wichtigkeits-Karte eine rudimentäre Form von Neugier simuliert werden kann: Unter der Voraussetzung, dass ein Erinnerungsvermögen des Gesehenen existiert, könnte die Wichtigkeits-Karte das zuvor Gesehene berücksichtigen - z.B. dadurch, dass die zuvor als am wichtigsten im Bild bewerteten Bildbereiche von der aktuellen Wichtigkeits-Karte abgezogen werden und erst im Anschluss daran der Schwellwert angewandt wird. So ließe sich verhindern, dass fortlaufend der selbe Bildbereich als das Wichtigste erkannt würde, was in einem Absuchen des gesamten Sichtfeldes resultieren sollte: Bei jedem neuen Betrachten gibt es mehr Bildbereiche, die bereits betrachtet wurden, sodass ein anderer als wichtig eingestuft wird.

Dieses Absuchen könnte sogar als Anlass für Verhalten genutzt werden, das über die visuelle Verarbeitung hinaus geht: Ist das ganze Sichtfeld abgesucht, so könnte dies ein Signal dafür sein, dass die Umgebung erkundet werden sollte, oder für anderweitige Selbstbeschäftigung. Der Grund für die noch fehlende Realisierung dieses Verhaltens ist, dass die Implementierung durch die Möglichkeit verschiedener Zoom-Stufen komplexer wird; gleiches gilt für das hier angenommene Erinnerungsvermögen.

Ein sehr rudimentäres automatisches Vergrößern von als wichtig bewerteten Bildbereichen wurde testweise bereits implementiert, jedoch ist die Verwendung nicht sinnvoll, solange kein Erinnerungsvermögen vorliegt und die extrahierten Linien nicht erkannt werden können. Erst mit implementierter Symbolerkennung kann eine Abbruchbedingung realisiert werden - ohne diese wird nur ziellos vergrößert. Ist die Erkennung gegeben, so kann *wahrscheinlich* auf Basis des Ergebnisses eine klarere Entscheidung über die nötigen Veränderungen am Bild getroffen werden, allerdings lässt sich das ohne jegliche Experimente nicht mit Sicherheit sagen.

Es sollte ebenfalls betont werden, dass die Wichtigkeits-Karte prinzipiell um zusätzliche Informationen erweitert werden kann, welche semantisch von statischen Bildern abweichen: Es könnte bspw. eine Bewegung im Bild automatisch als wichtiger bewertet werden, als unbewegte Bildanteile; die Wahrnehmung der Bewegung könnte durch Differenzbilder erfolgen. Eine mögliche Implementierung wäre es, am Rand des Sichtfeldes eine Bewegung höher zu bewerten, während in der Bildmitte mehr Fokus auf Farben gelegt wird, ähnlich wie es bei der menschlichen Wahrnehmung der Fall ist.

Während diese Priorisierung von Bewegtem im Umgang mit Myon allgemein von Nutzen sein könnte, da so die „Aufmerksamkeit“ von außerhalb lenkbar wäre, erfordert ein Nutzen im Kontext der Symbolerkennung erneut ein Erinnerungsvermögen; ist dies gegeben, so könnte bspw. mit einem Laserpointer eine Form auf einen statischen Hintergrund „gezeichnet“ werden - die Realisierbarkeit ist jedoch unklar und dieses Beispiel damit rein spekulativ.

Neben diesen größeren Verbesserungen sind kleinere Erweiterungen möglich wie bspw. ein leichtes Hin-und-Her-Verschieben des betrachteten Bildbereichs (den Mikro-Sakkaden im menschlichen Auge ähnelnd) sowie die Verbesserung von bereits implementierten Schritten, insbesondere des verwendeten Tiefpassfilters.

4 Ergebnisse

Das vorgestellte Verfahren wurde in der Erkennung von einzelnen Symbolen, Buchstaben und einfachen elektrischen Schaltplänen getestet. Dabei wurde der Hintergrund (Whiteboard, weißes Papier, kariertes Papier, Tablet-Bildschirm), die Schriftfarbe (Schwarz und Rot) und teils auch die Liniendicke variiert, um verschiedene Szenarien zu testen. Im Folgenden werden einzelne Ergebnisse aus diesen Tests vorgestellt. Die Darstellungskonvention aus Abb. 15 wird hier fortgeführt: Im Hintergrund ist die Wichtigkeits-Karte nach der Anwendung des Schwellwerts zu sehen, um die Linienbildung nachvollziehbar zu halten, darüber wurden die erkannten Linien gezeichnet. Die im Folgenden vorgestellten Aufnahmen entstanden zu verschiedenen Tageszeiten und Wetterverhältnissen, was sich auf die Lichtverhältnisse auswirkte, auch wenn versucht wurde, direkte Reflexionen des Himmels in den betrachteten Bildausschnitten zu vermeiden. Alle Testbilder waren ca. 40 cm von Myons Kamera entfernt.

4.1 Linienerkennung in Buchstaben aus dem MNIST-Datensatz

Um einen ungefähren Vergleich zu einer der vorgestellten Stroke Extraction-Methoden ziehen zu können, wurden verschiedene Buchstaben aus dem MNIST-Datensatz (2017) [20], einer großen Sammlung von Hand geschriebener alphanumerischer Symbole, verwendet - L'Homer gibt in seinem Paper ein Beispiel seines Algorithmus auf Basis eines nicht weiter spezifizierten NIST-Datensatzes. Bei der Auswahl der hier verwendeten Bilder wurde versucht, Exemplare zu finden, die denen entsprechen, die L'Homer verwendete; auf Grund der Größe des Datensatzes ist die Übereinstimmung jedoch nur zum Teil gegeben. Diese Bilder wurden dann auf dem Bildschirm eines Tablets angezeigt, welcher von Myons Kamera erfasst wurde - es ist also kein direkter Vergleich, da das Bild nicht einfach binär in den Algorithmus eingespeist werden konnte. Die Auflösung ist ebenfalls nicht dieselbe: Die Bilder des MNIST-Datensatzes liegen in einer Auflösung von 128×128 Pixeln vor, nicht den 22×22 Superpixeln, auf denen hier gearbeitet wurde. (Eine Darstellung des Unterschieds erfolgt in Abb. 17.)

Wie in Abb. 16 zu sehen ist, hat die verwendete Auflösung/Vergrößerung des Bildes einen starken Einfluss auf die erkannten Linien:

Wird der Buchstabe mit vielen (Super-)Pixeln angezeigt („Zoom: 3“), so wird der ungefähre Umriss des Buchstabens zu Linien umgewandelt, auch wenn dieser Umriss teils unterbrochen ist. Nimmt der Buchstabe jedoch vergleichsweise wenige Pixel ein („Zoom: 6“), so werden weniger Linien erkannt, welche jedoch ungefähr die Form des Buchstabens beschreiben.

Dieses Muster, dass die richtige Zoom-Stufe gefunden werden muss, um das beste Ergebnis zu erzielen, findet sich in all den hier vorgestellten Ergebnissen wieder. In Abs. 3.2 wurde ein Verhalten zum automatischen Finden dieses Optimums ansatzweise betrachtet.

In beiden Fällen sowie bei den dazwischenliegenden Zoom-Stufen werden auch unerwünschte Linien erkannt. Zum Umgang mit diesen unerwünschten Linien wurden in Abs. 3.2 Möglichkeiten aufgezeigt. Würden diese verwendet, so könnte das „Zoom: 6“ Ergebnis aus Abb. 16 wahrscheinlich erfolgreich zur Erkennung verwendet werden.

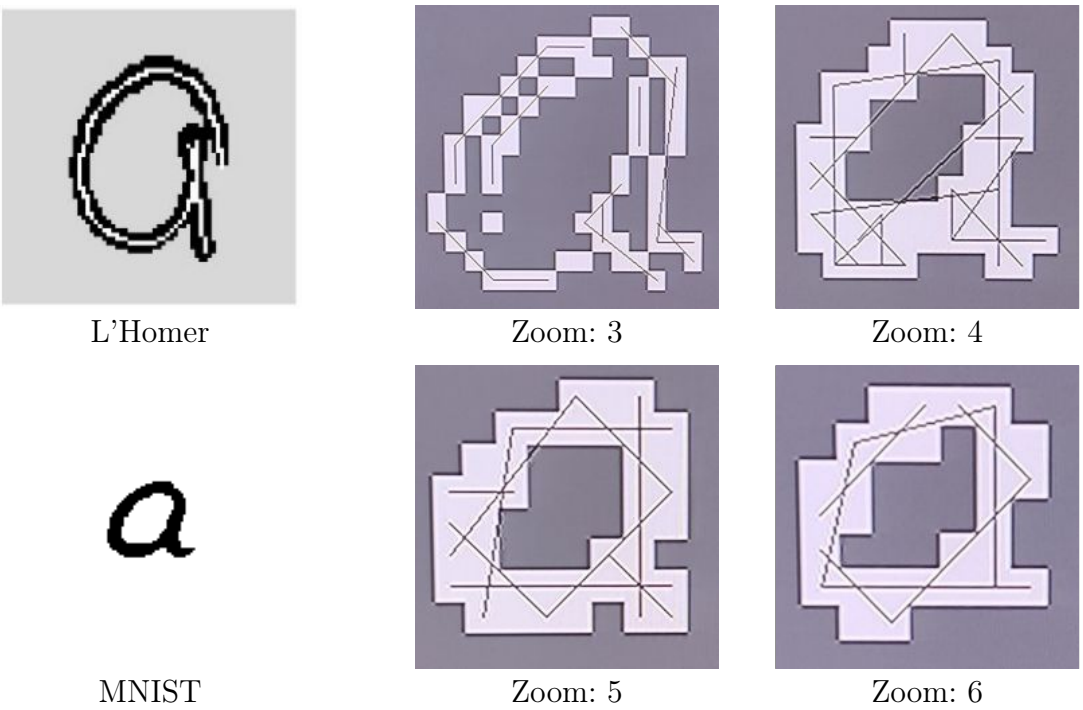


Abbildung 16: Vergleich mit L'Homers Stroke Extraction: Links ist ein Ergebnis von L'Homer und ein ungefähr ähnlicher Buchstabe aus dem MNIST-Datensatz abgebildet. Rechts sind vier Ergebnisse des in dieser Arbeit entwickelten Algorithmus bei verschiedenen Zoom-Stufen. Dabei entspricht eine kleinere Zahl einer stärkeren Vergrößerung.[9]

Weitere Vergleiche sind in Abb. 17 und 18 dargestellt: In Abb. 17 wird der Unterschied zwischen dem Bild aus dem MNIST-Datensatz und dem tatsächlich verwendeten Input in den Erkennungsalgorithmus hervorgehoben und das Ergebnis vollständig gezeigt, um auch den Einfluss des Lichts zu zeigen, während in Abb. 18 zusätzliche Vergleiche zu der Stroke Extraction L'Homers gegeben sind; hier ist das Ergebnis zur besseren Sichtbarkeit vergrößert worden.

[9]Das angegebene Level des Zooms ist immer die Kantenlänge eines Superpixels in Pixeln - je kleiner die Zahl, desto stärker die Vergrößerung. Genaueres zur PixelPipeline wurde in Abs. 1.1.1 aufgeführt, Superpixel wurden in Abs. 1.1.2 erklärt.

MNIST | Kamera Input | Erkennung Input | Erkennung Output

Abbildung 17: Zwischenergebnisse im Erkennen von MNIST-Bildern. Von links nach rechts: Das Bild aus dem MNIST-Datensatz, ein Foto des ungefähren Blickwinkels von Myons Kamera, Ausgabe der PixelPipeline und ein Ergebnis dieses Algorithmus.

In Abb. 17 ist der Unterschied zwischen dem Eingangsbild, auf Basis dessen die Linienerkennung durchgeführt wird, klar zu sehen - nicht nur ist das Bild, welches für den in dieser Arbeit behandelten Algorithmus verwendet wird, sehr viel verpixelter, es kommen außerdem Reflexionen hinzu; um einen echten Vergleich zu ziehen, müssten die Binärbilder direkt verwendet werden.

Es ist dennoch anzunehmen, dass L'Homers Ansatz für die Stroke Extraction dem hier vorgestellten überlegen ist - zumindest was die Genauigkeit angeht - da keine zusätzlichen Linien erkannt werden. Weder ein Vergleich der Laufzeit noch der Komplexität war möglich, da diese Informationen in seinem Paper nicht aufgeführt sind. Auf die Laufzeit des hier vorgestellten Algorithmus wird in Abs. 4.5 eingegangen.

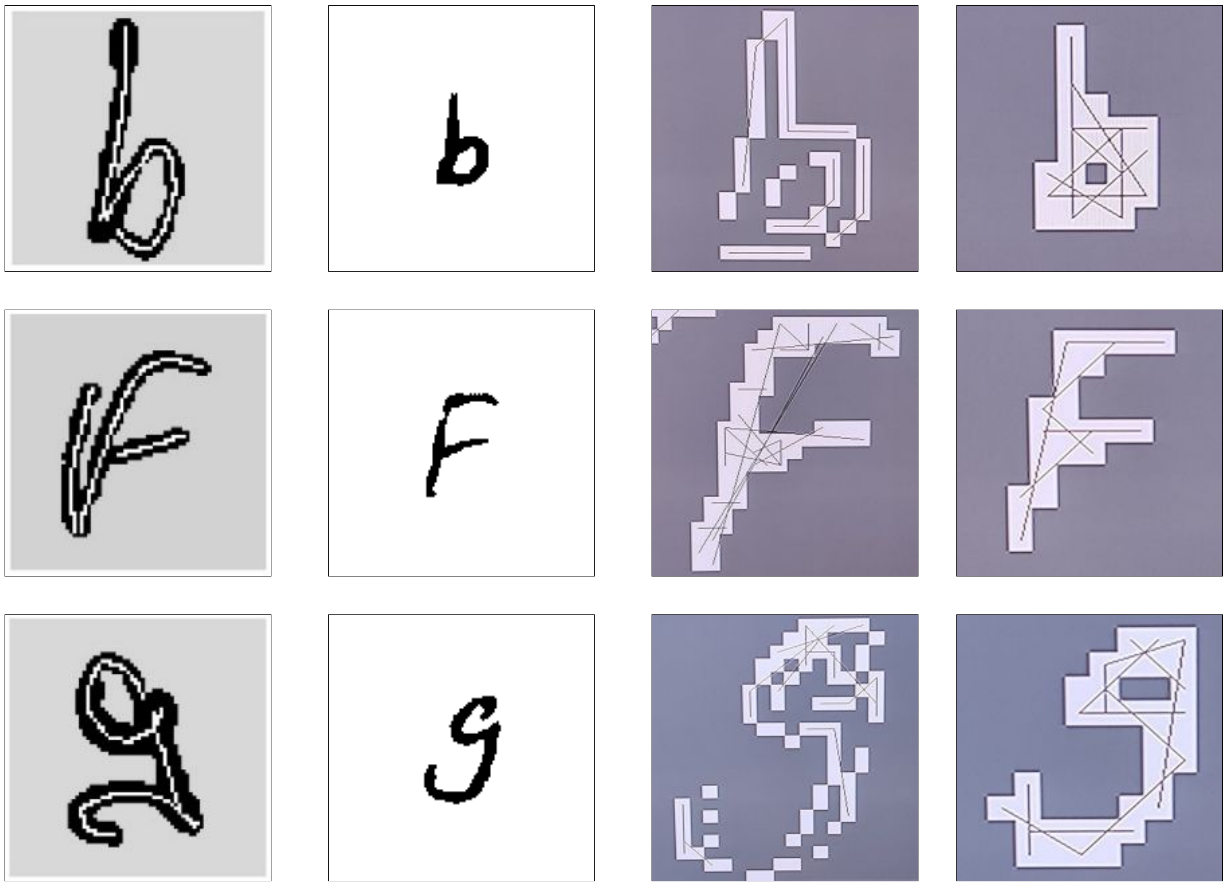
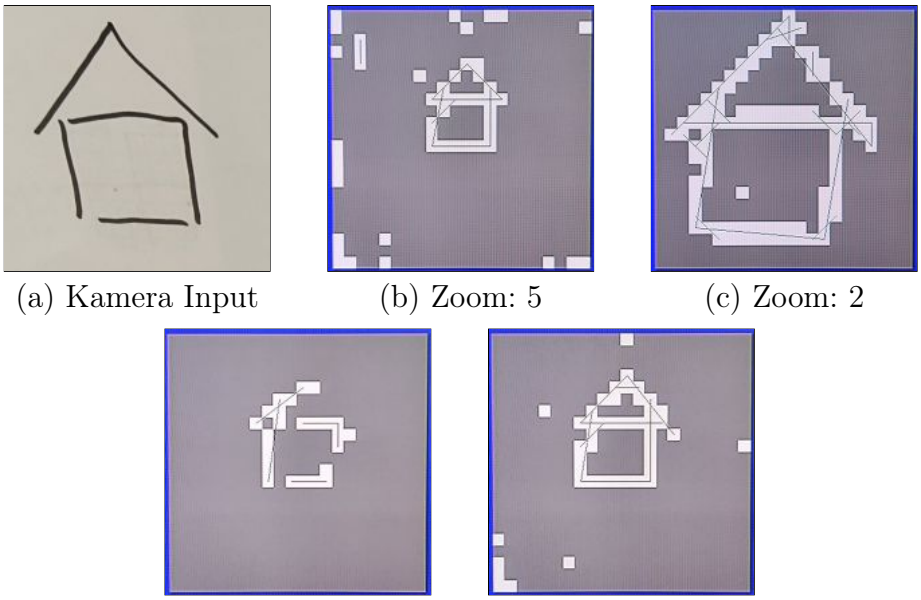


Abbildung 18: Weiterer Vergleich mit L'Homers Stroke Extraction. Von links nach rechts: Ein Ergebnis von L'Homers Stroke Extraction, ein grob entsprechendes Bild aus dem MNIST-Datensatz und zwei Ergebnisse dieses Algorithmus.

4.2 Linienerkennung in Strichzeichnungen

Da Buchstabenerkennung nicht das eigentliche Ziel der Arbeit war, wird nun jedoch der Fokus auf die Erkennung von Strichzeichnungen gelenkt, welche typischerweise aus halbwegs geraden Linien bestehen und meist nicht so kompakt wie Buchstaben sind. Wie in den folgenden Testergebnissen zu sehen ist, ist die Linienerkennung bei Strichzeichnungen meist besser als bei den zuvor betrachteten Buchstaben.

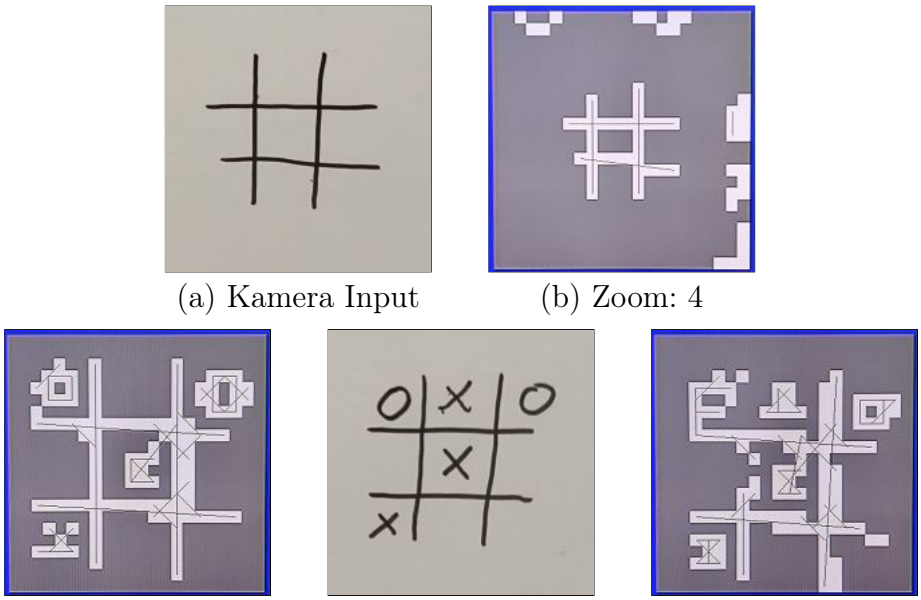
Die größten Probleme treten auf, wenn sich Linien zu nahe kommen und als zusammengehörig erkannt werden - dies war bei den durchgeführten Tests jedoch nicht sehr häufig der Fall. Aus Gründen der Übersichtlichkeit wird in folgenden Abbildungen zumeist auf das Ausgabebild der PixelPipeline verzichtet.



(a) Kamera Input (b) Zoom: 5 (c) Zoom: 2

(d) Zoom: 4 (e) Zoom: 4

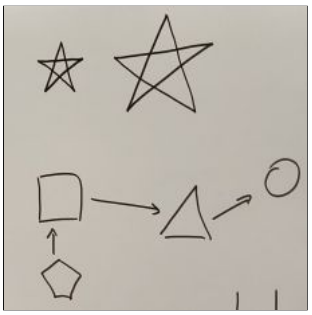
Abbildung 19: Linienerkennung an einem einfach gezeichneten Haus auf weißem Papier. Während die Erkennung bei (b) bereits passend ist, war diese nicht stabil, wie mit (d) und (e) verdeutlicht wird: Mit einer leichten Verschiebung des betrachteten Bereichs kann statt dem schlechten Match (d) die sehr viel bessere Linienerkennung (e) erreicht werden. Die in (c) abgebildete Erkennung war dagegen auch bei angewandter Verschiebung stabil.



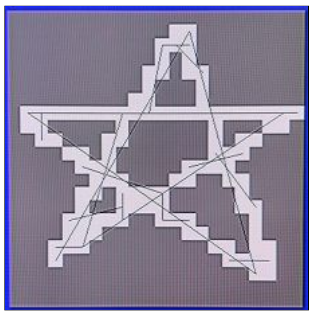
(a) Kamera Input (b) Zoom: 4

(c) Zoom: 2 (d) Kamera Input (e) Zoom: 2

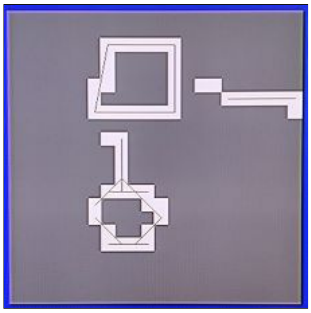
Abbildung 20: Linienerkennung für Tic-Tac-Toe auf weißem Papier. Die Linien werden beim leeren Spielfeld (b) gut erkannt, beim Ausfüllen kann es jedoch zu Schwierigkeiten kommen - eine Erinnerung an das bisherige Spiel oder ein Anpassen des betrachteten Bildbereichs wäre für eine mögliche Interaktion wahrscheinlich von Nöten, da die Symbole auch bei weiterer Vergrößerung nicht immer erkennbar sind und teils mit dem Spielfeld verschmelzen (e).



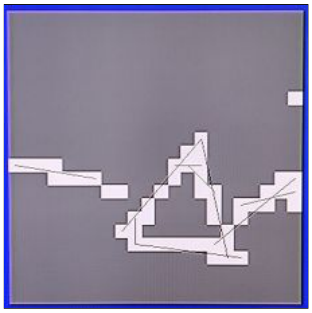
(a) Kamera Input



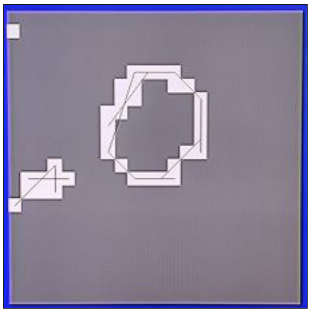
(b) Zoom: 2



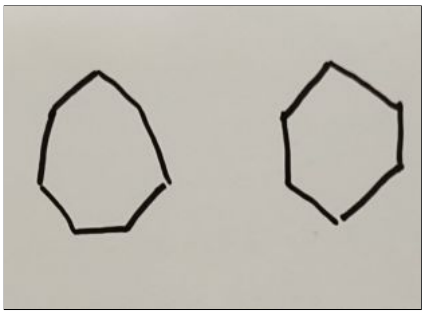
(c) Zoom: 4



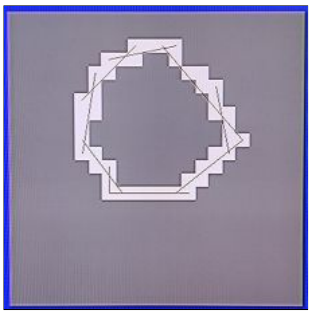
(d) Zoom: 3



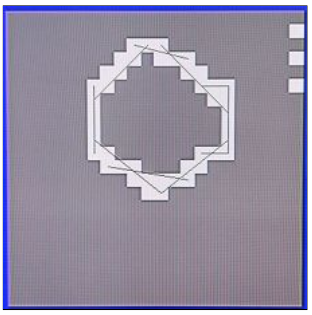
(e) Zoom: 2



(f) Kamera Input



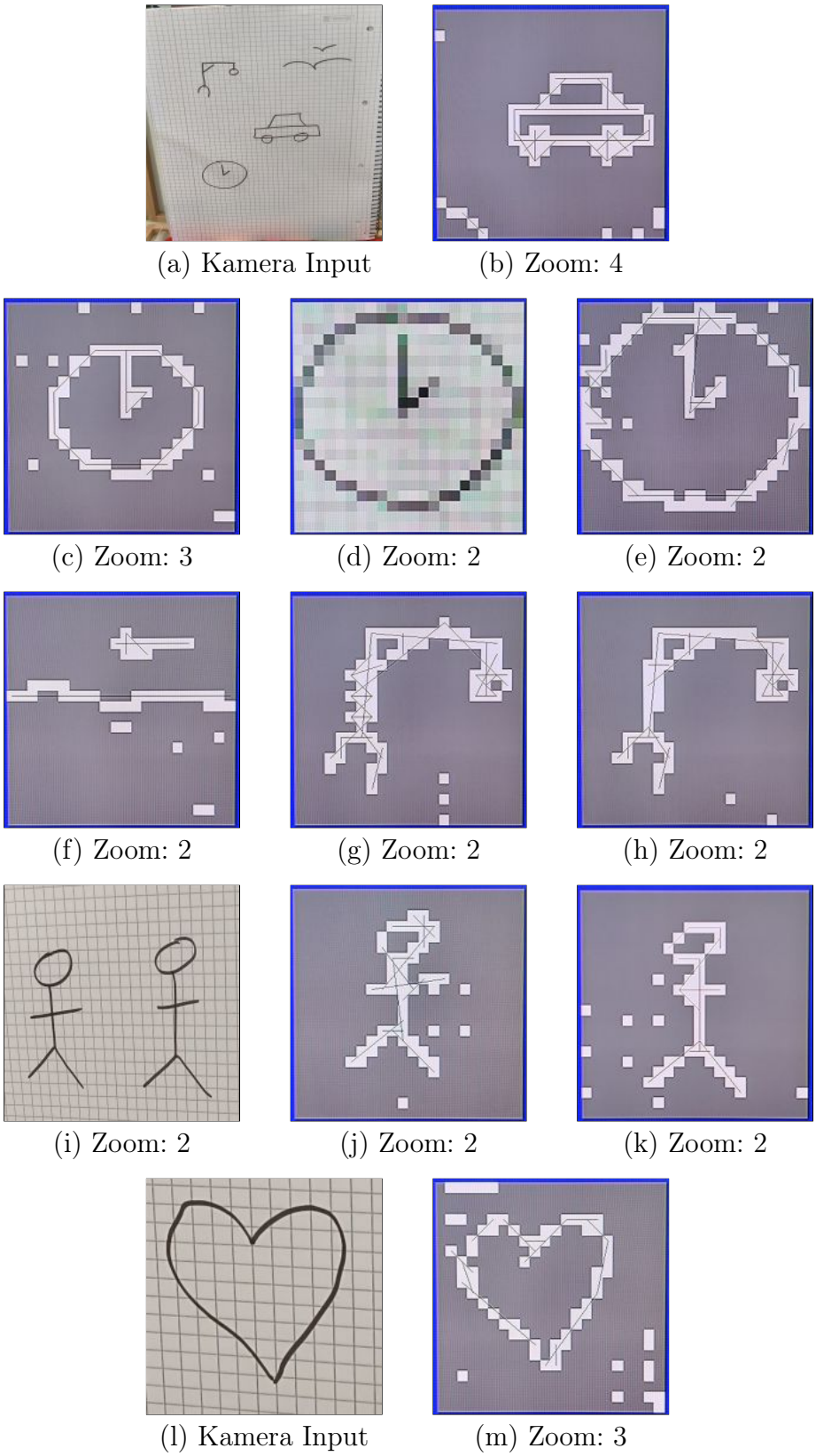
(g) Zoom: 2



(h) Zoom: 2

Abbildung 21: Linienerkennung für einfache Symbole auf weißem Papier. Es fällt auf, dass die Ergebnisse der Linienerkennung für reguläre Polygone einander mit zunehmenden Innenwinkeln ähnlicher werden; der Kreis (e) kann ggf. noch vom Pentagon (c) unterschieden werden, bei dem Hexa- (h) oder Heptagon (g) wird dies jedoch vermutlich zu unzuverlässig sein. Bei dem Pentagramm (b) handelt es sich um das größere der beiden in (a) abgebildeten.

4.2.1 Linienerkennung auf kariertem Papier



(a) Kamera Input (b) Zoom: 4

(c) Zoom: 3 (d) Zoom: 2 (e) Zoom: 2

(f) Zoom: 2 (g) Zoom: 2 (h) Zoom: 2

(i) Zoom: 2 (j) Zoom: 2 (k) Zoom: 2

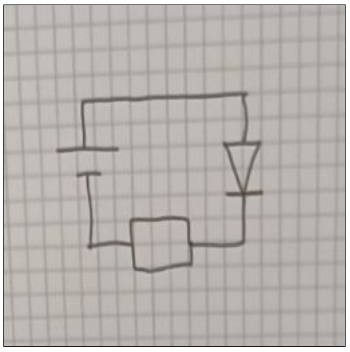
(l) Kamera Input (m) Zoom: 3

Abbildung 22: Linienerkennung auf kariertem Papier. Die Symbole wurden ausgewählt, um für Interaktionen potentiell relevante Symbole zu testen - der Galgen ist für das Spiel „Galgenmännchen“ gedacht. Obwohl das Gittermuster des Papiers bei ausreichender Vergrößerung selbst nach der Vorverarbeitung durch die PixelPipeline noch klar sichtbar ist (d), sind die Linien in den meisten Bildern gut erkannt worden. Nur bei (g) hatte das Muster tatsächlich einen relevanten Effekt auf die erkannten Linien, was sich durch Verschieben (vgl. Abs. 4.4.1) lösen ließ (h).

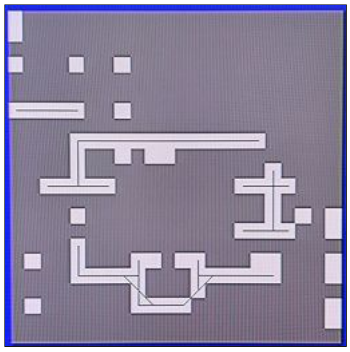
4.3 Linienerkennung in elektrischen Schaltplänen

Während die Linienerkennung in Strichzeichnungen gut funktioniert hat, ist die Erkennung von Bauelementen in elektrischen Schaltplänen nur vereinzelt erfolgreich gewesen. Das größte Problem scheint erneut zu sein, dass Linien miteinander verschmelzen, allerdings passiert dies in Schaltplänen anscheinend häufiger, wie in den nachfolgenden Abbildungen zu sehen ist: In Abb. 23 gab es Probleme mit dem Gittermuster des Papiers und in Abb. 24 wurde die Beschriftung der Widerstände und des Operationsverstärkers mit den Linien der Bauteile verbunden.

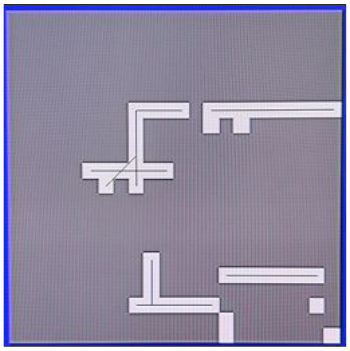
Es ist beim Verfassen dieser Arbeit noch unklar, ob die geplanten Verbesserungen, welche in Abs. 3.2 beschrieben wurden, auch für das Bewältigen *dieses* Problems ausreichen werden.



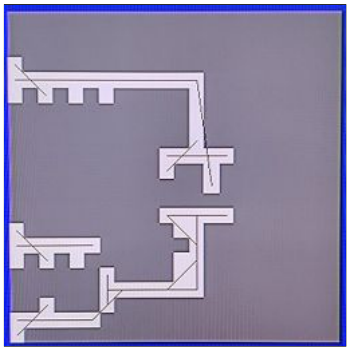
(a) Kamera Input



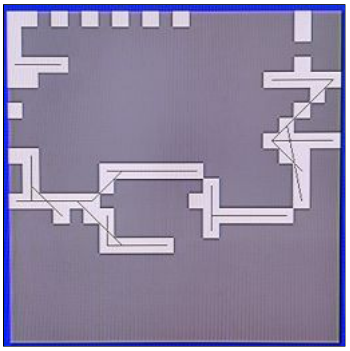
(b) Zoom: 3



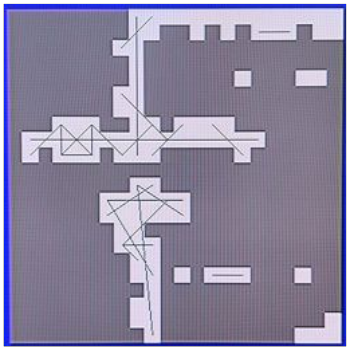
(c) Zoom: 2



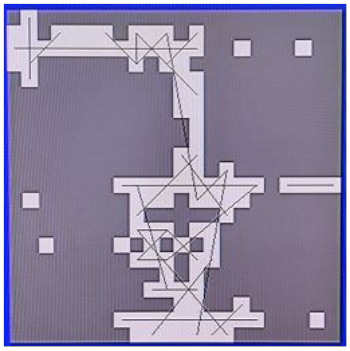
(d) Zoom: 2



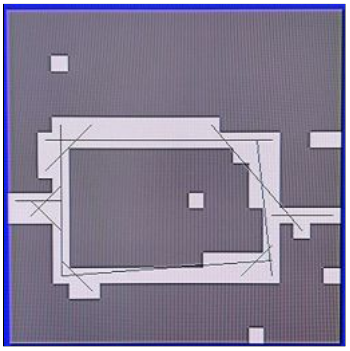
(e) Zoom: 2



(f) Zoom: 1

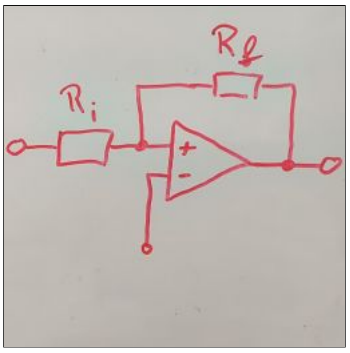


(g) Zoom: 1

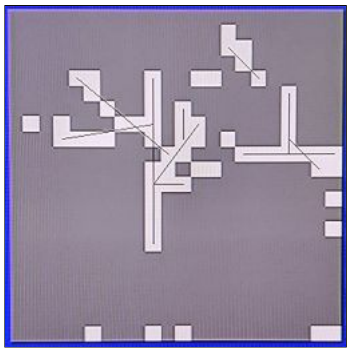


(h) Zoom: 1

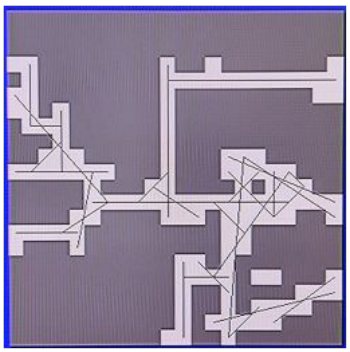
Abbildung 23: Linienerkennung bei einem simplen, unbeschrifteten Schaltplan auf kariertem Papier. Die Diode (g) und die untere Hälfte der Spannungsquelle (f) sind kaum erkennbar, während der Widerstand (h) vergleichsweise gut abgebildet ist.



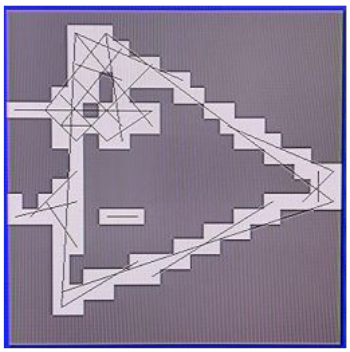
(a) Kamera Input



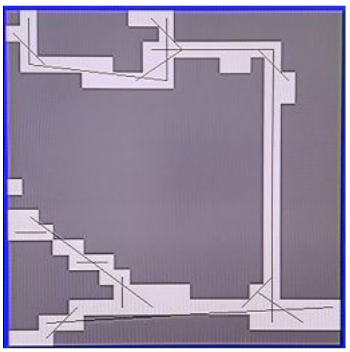
(b) Zoom: 7



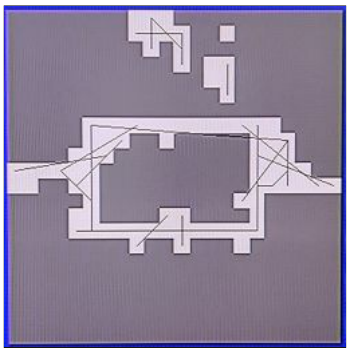
(c) Zoom: 3



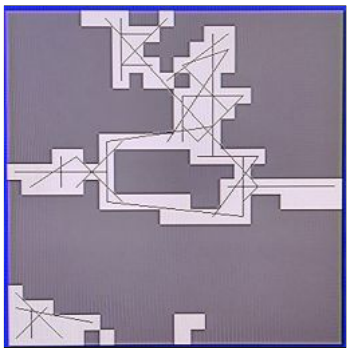
(d) Zoom: 2



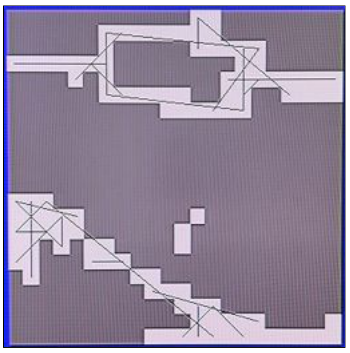
(e) Zoom: 2



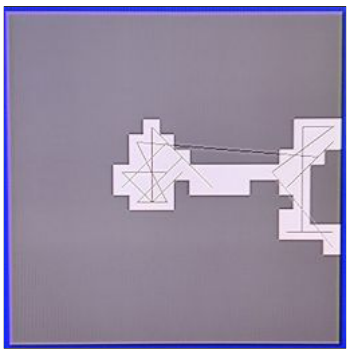
(f) R_i Zoom: 2



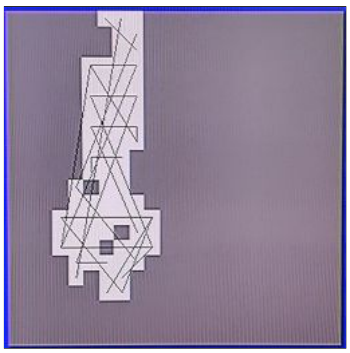
(g) R_f Zoom: 2



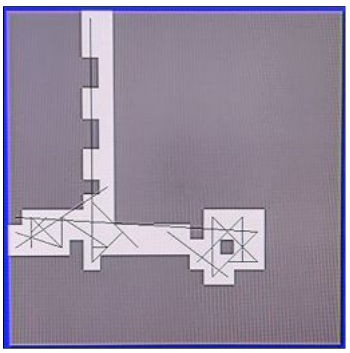
(h) R_f Zoom: 2



(i) Zoom: 2



(j) Zoom: 1



(k) Zoom: 2

Abbildung 24: Linienerkennung bei einem Schaltplan auf einem Whiteboard. Die Erkennung der Bauteile ist weitestgehend besser als in Abb. 23, allerdings sind mehr unerwünschte Linien vorhanden. Teils verschmilzt die Beschriftung mit den Bauteilen, wie in (g) dargestellt. Während sich dies durch isolierte Betrachtung lösen lässt (h), ist in diesem Fall ein größerer Abstand wahrscheinlich erforderlich. Ähnlich ist es bei dem + des OP in (d). Bilder mit Zoom-Stufe 1 sind bei dieser Liniendicke meist zu chaotisch, um damit arbeiten zu können, wie in (j) zu sehen ist.

4.4 Erkennbarkeit der re-synthetisierten Strichzeichnungen

Da die menschliche visuelle Verarbeitung sehr komplex ist, kann es schwierig sein, die tatsächliche Erkennbarkeit der abgebildeten Symbole nur auf Basis der eingezeichneten Linien zu bewerten, wenn im Hintergrund noch die Wichtigkeits-Karte zu sehen ist. Während sie in vorigen Abschnitten mit abgebildet wurde, um den Vergleich mit dem Kamera Input zu erleichtern, wird in diesem Abschnitt deshalb bewusst darauf verzichtet, sodass die Erkennbarkeit besser eingeschätzt werden kann.

Mit der gleichen Motivation werden auf den folgenden zwei Seiten zumeist einander ähnliche Symbole und Bilder betrachtet, sodass bewertet werden kann, ob diese tatsächlich erkennbar und voneinander unterscheidbar sind. Es werden links die Eingangsbilder und rechts die re-synthetisierten Strichzeichnungen dargestellt, sodass ein Selbsttest (durch Verdecken der linken Seitenhälfte) leichter möglich ist.

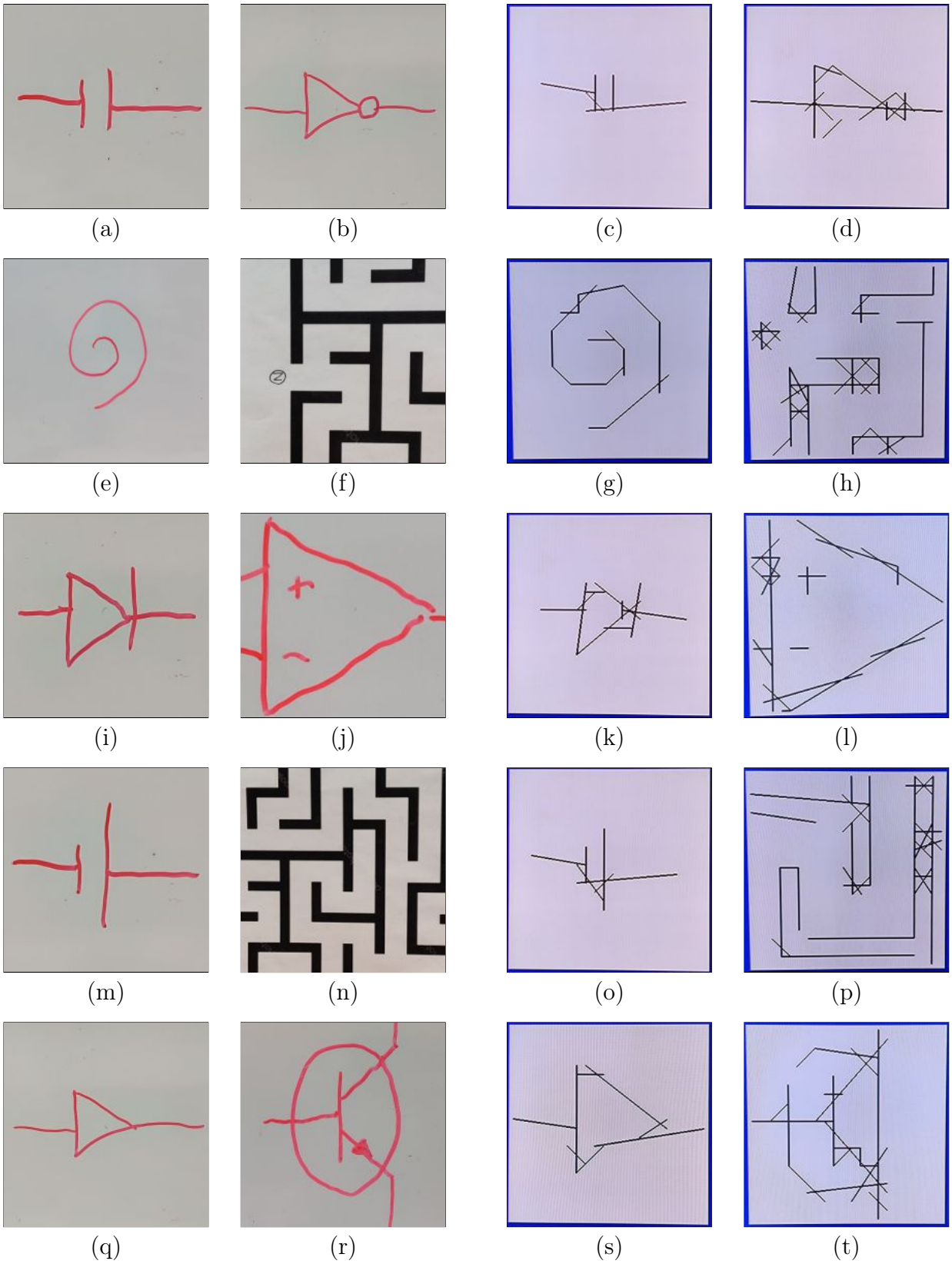


Abbildung 25: Beispiel für die Erkennbarkeit von re-synthetisierten Strichzeichnungen. Die jeweils linke Strichzeichnung entspricht dem jeweils linken Originalbild, also gehört (c) zu (a), (d) zu (b), etc. Aus den gemachten Aufnahmen wurden (sofern vorhanden) gezielt diejenigen Bilder ausgewählt, welche für die Erkennung ungünstig erschienen. Günstigere Varianten dieser Bilder (sowohl aus dieser als auch aus der nachfolgenden Abbildung) sind im Appendix in Abb. 36 gegeben.

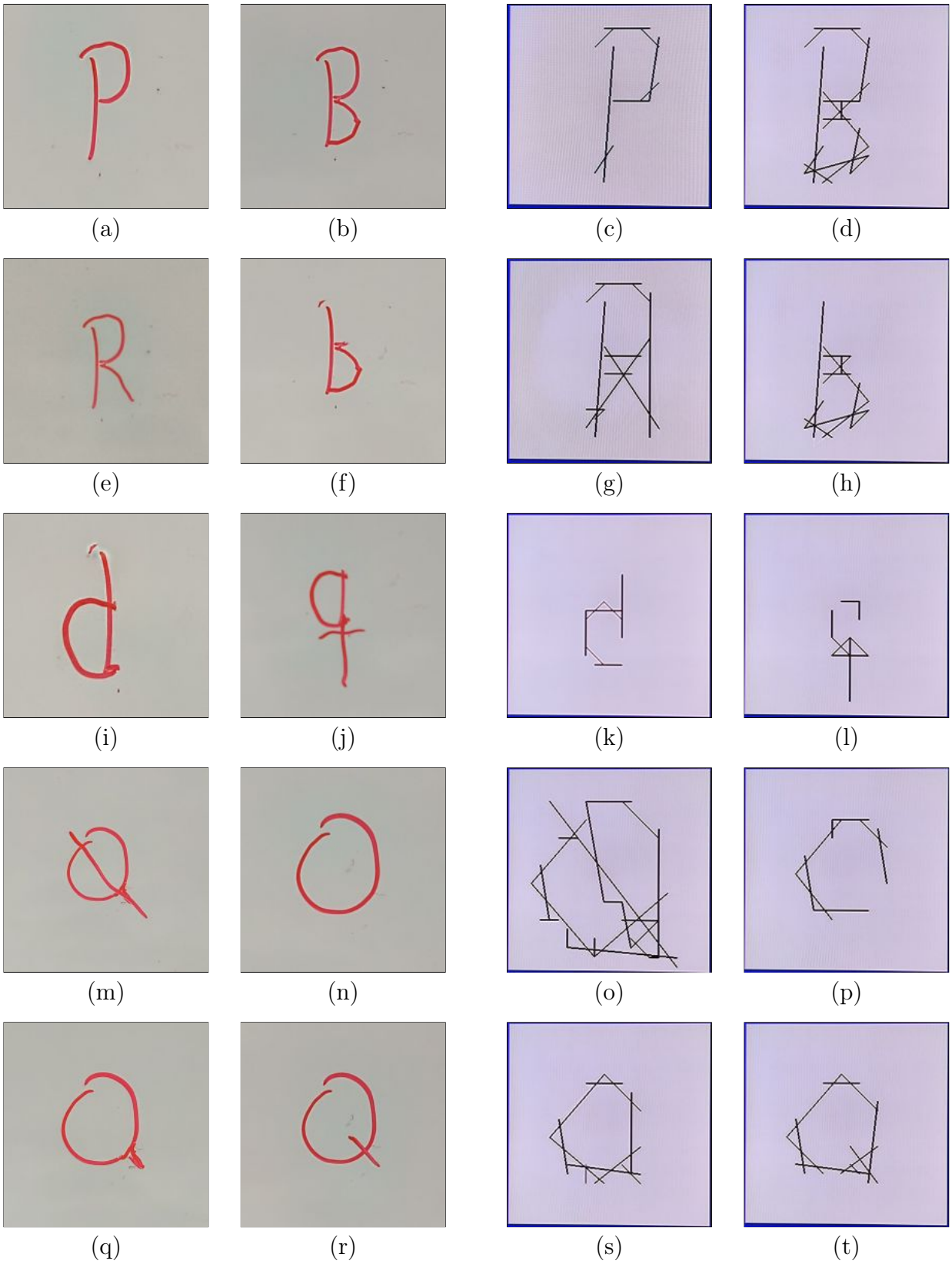


Abbildung 26: Weiteres Beispiel für die Erkennbarkeit von re-synthetisierten Strichzeichnungen. Die jeweils linke Strichzeichnung entspricht dem jeweils linken Originalbild, also gehört (c) zu (a), (d) zu (b), etc. Aus den gemachten Aufnahmen wurden (sofern vorhanden) gezielt diejenigen Bilder ausgewählt, welche für die Erkennung ungünstig erschienen. Günstigere Varianten dieser Bilder (sowohl aus dieser als auch aus der vorherigen Abbildung) sind im Appendix in Abb. 36 gegeben.

4.4.1 Variabilität in erkannten Linien

Wie zuvor nur erwähnt wurde, kann bei gleichem Kamerawinkel die selbe Strichzeichnung selbst bei gleichbleibender Zoom-Stufe unterschiedlich erkannt werden. Diese Variabilität kann auf zwei Ursachen zurückgeführt werden:

Zum einen wirkt sich eine leichte Verschiebung des betrachteten Bildbereichs auf die erkannten Linien aus. Dies liegt daran, dass sich die Farbe einzelner Superpixel durch diese Verschiebung ändert und so eine andere Wichtigkeits-Karte erzeugt wird, da die den Superpixeln zugehörigen Pixel nach der Verschiebung anders gruppiert werden.

Die andere Ursache ist Rauschen im Bild; es wird vermutet, dass die Quelle des Rauschens das Flackern künstlicher Lichtquellen ist. Durch dieses Rauschen können selbst ohne jegliche Änderung des Bildbereichs unterschiedliche Linien erkannt werden. In beiden Fällen ändert sich das Erkannte nicht willkürlich sondern in einem wiederkehrenden Muster. Ein Beispiel ist in Abb. 27 gegeben.

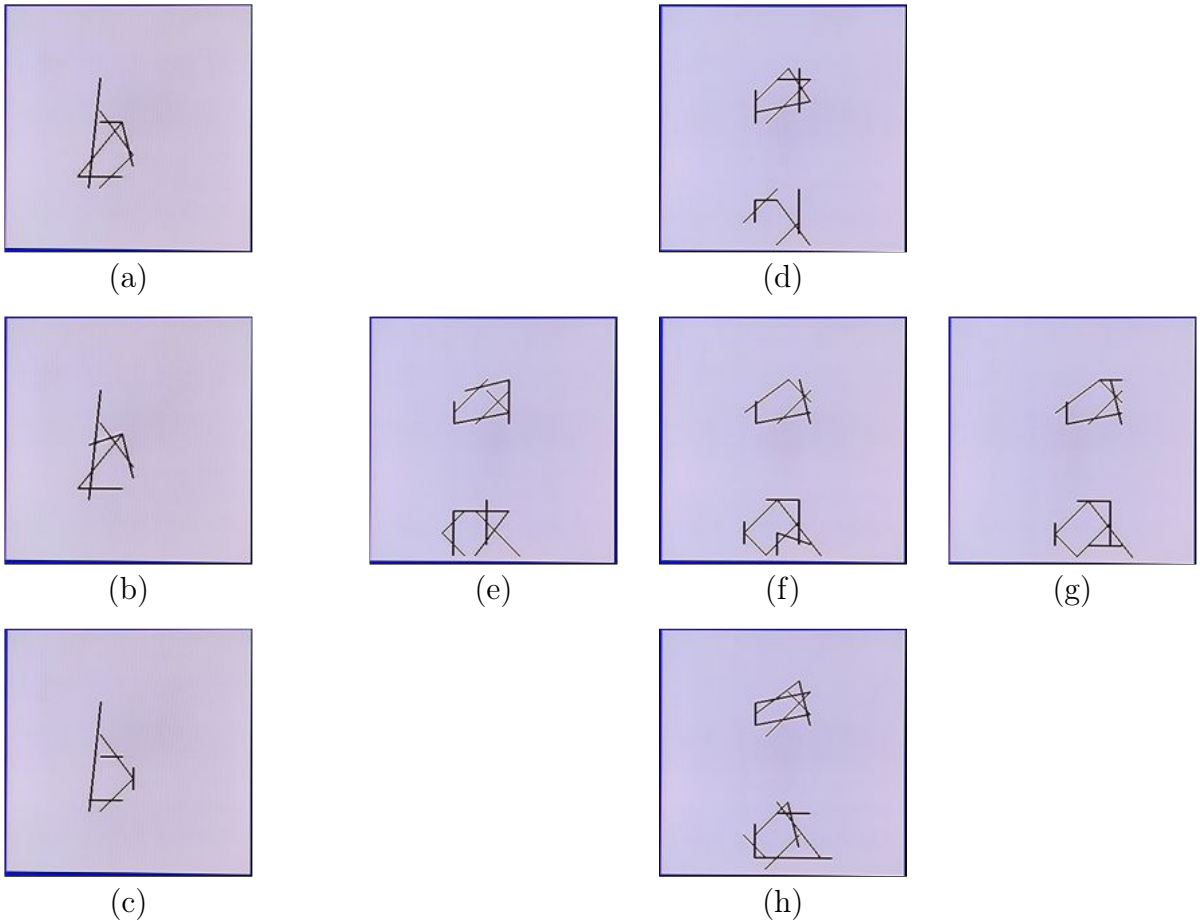


Abbildung 27: Darstellung von Variabilität in der Linienerkennung. Abbildungen (a-c) zeigen die drei Varianten eines handschriftlichen `'b'`, die bei einer Aufnahme ohne jegliche Verschiebung des betrachteten Bildbereichs auftraten. Abbildungen (d-h) zeigen den Effekt, den leichtes Verschieben auf zwei übereinander liegende und voneinander distinkte Schreibweisen eines `'a'` hatte. In beiden Fällen sind alle bei den entsprechenden Aufnahmen auftretenden Varianten aufgeführt - weiteres Warten (im Falle von a-c) oder Verschieben (d-h) hatte lediglich Wiederholungen bereits erkannter Linienkombinationen zur Folge.

4.5 Laufzeit

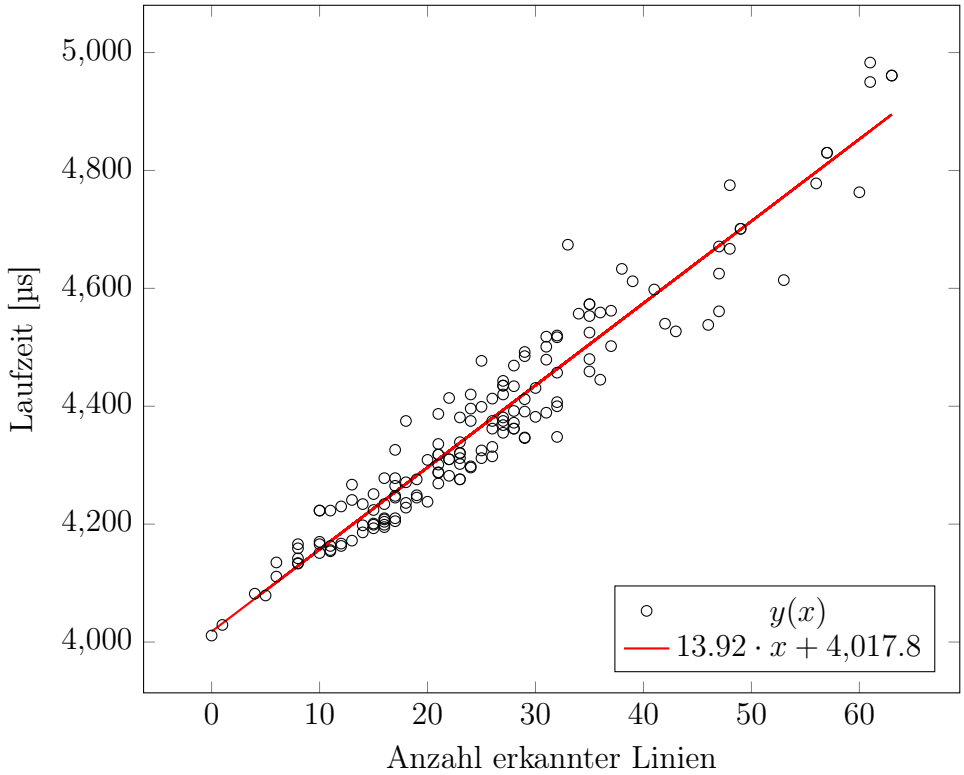


Abbildung 28: Laufzeitverhalten des vorgestellten Algorithmus. Die Laufzeit wurde für den Algorithmus ohne die Anzeigeroutinen gemessen. Die mit `pgfplots` erzeugte lineare Regressionsgerade ist rot eingezeichnet. Die aktuelle Implementierung des Algorithmus ist noch nicht optimiert worden - frühere Tests lassen die Annahme zu, dass ca. 1000 μs von dem noch sehr simpel implementierten Tiefpassfilter verwendet werden. Es wird erwartet, dass das Tiefpassfilter um einiges effizienter gestaltet werden kann, was sich positiv auf die Laufzeit des Algorithmus auswirken sollte, ohne dass das Verhalten verändert wird.

Während nicht feststeht, dass die Laufzeit des Algorithmus verlässlich mit der Anzahl der erkannten Linien korreliert, lässt Abb. 28 diesen Schluss zu. Das zu der eingezeichneten linearen Regressionsgerade berechnete Bestimmtheitsmaß ist $R^2 = 0.928$, jedoch ist die Aussagekraft in dem erreichten Wertebereich sehr begrenzt; das Bestimmtheitsmaß einer exponentiellen Trendlinie passt mit $R^2 = 0.929$ minimal besser, doch wurden bei den durchgeführten Tests nie mehr als 63 Linien in einem Bild erkannt und eine sichere Aussage über den Trend lässt sich auf Basis dieser Datenlage nicht treffen.

Gleichzeitig ist die Tatsache, dass nie mehr als 63 Linien erkannt wurden, ein Indiz dafür, dass nur selten sehr viele Linien erkannt werden; die für Abb. 28 erhobenen Daten unterstützen diese Beobachtung, allerdings müssten für eine verlässliche Aussage ggf. umfangreichere Tests durchgeführt werden. Für die geplante Anwendung auf dem Roboter Myon ist eine maximale Dauer von ca. 5 ms akzeptabel, jedoch nicht optimal. Wie in Abs. 1.1.1 erwähnt wurde, liegt ein neues Bild alle 20 ms vor, der Algorithmus kann also problemlos auf jedes eingehende Bild angewandt werden. Allerdings würde in einem Anwendungsszenario nicht nur dieser Algorithmus aktiv sein, weshalb die Laufzeit möglichst gering gehalten werden sollte. Es wird erwartet, dass die Laufzeit durch die Optimierung bereits bekannter Problemstellen um ca. 1 ms reduziert werden kann; wie viel mehr möglich ist, ist noch unklar.

4.6 Vergleich des implementierten Algorithmus zum menschlichen Sehen

In Abs. 1.2 wurde das menschliche Sehen als Grundlage und Inspiration eingeführt. In diesem Abschnitt werden die Unterschiede und Gemeinsamkeiten zwischen dem in dieser Arbeit entwickelten Algorithmus und dem menschlichen Sehen dargelegt.

In der Kamera und PixelPipeline Myons wird das eingehende Bild analog zu der menschlichen Retina verarbeitet: Die Kamera entspricht dabei der Linse und den Photorezeptorzellen während die PixelPipeline die Rolle der rezeptiven Felder übernimmt.

Ein klarer Unterschied liegt in der Verarbeitung des einfallenden Lichts: Die Fovea Centralis des menschlichen Auges hat eine sehr viel höhere Auflösung als die Kamera Myons, während die Auflösung in der Peripherie vergleichbar oder geringer ist. Die Farbtiefe und der mögliche Farbraum sind im menschlichen Auge ebenfalls umfangreicher und die Photorezeptorzellen besitzen eine logarithmische Impulsantwort auf Helligkeit, während die Kamera linear reagiert. Mit den ersten Schritten des hier vorgestellten Algorithmus wird versucht, diesen Unterschied zu reduzieren: Die Trennung der Farbdimensionen, Skalierung und Posterisation bewirken gemeinsam eine Betonung der extremeren Farbwerte, auch wenn dies nicht ausreicht, um diesen Unterschied vollständig auszugleichen.

Ein weiterer wichtiger Unterschied ist, dass die Signale der Photorezeptorzellen durch die rezeptiven Felder anders komprimiert werden, als es in der PixelPipeline geschieht: Beide fassen einen gegebenen Bildbereich zusammen - die rezeptiven Felder liegen jedoch zusätzlich in den beschriebenen ON-Center/OFF-Center-Varianten vor, was der erste Schritt der Kantenerkennung ist.

Die PixelPipeline ist in der Lage, den betrachteten Bildbereich alle 20 ms zu verändern; kleine Bewegungen des Auges (sog. Mikro-Sakkaden) erfolgen beim Menschen dagegen um einiges langsamer. Da Myon über nur eine Kamera verfügt, existiert kein Analog zum Chiasma Opticum und auch die LGN haben keine Entsprechung.

Die in der Retina begonnene Kantenerkennung wird im V1 weitergeführt und es entsteht daraus die Wichtigkeits-Karte. Analog erfolgt im Algorithmus nun die Kantenisolation mittels Erosion und die Rekombination der Farbkanäle zur Wichtigkeits-Karte.

Die Kantenerkennung im V1 ist lokal und Merkmal-spezifisch (Helligkeit, Farbe, Orientierung, simple Formen), weshalb die resultierende Wichtigkeits-Karte um einiges komplexer und genauer ist als die des vorgestellten Algorithmus.

Im Algorithmus erfolgt die Gewichtung der Kanten global und nur auf Basis der Häufigkeit von Helligkeit und Farben - Orientierung oder Formen werden dazu nicht betrachtet. Erst im Anschluss an die Erzeugung der Wichtigkeits-Karte können die vorliegenden Orientierungen mittels morphologischem Öffnen extrahiert werden, während dies im V1 über die Betrachtung des Merkmals „Orientierung“ ein Teil der Kantenerkennung ist.

Die erkannten Kanten bzw. Linien können daher im Algorithmus noch nicht in die Wichtigkeits-Karte einfließen, sodass häufigere Orientierungen nicht so unterdrückt werden können, wie es im V1 möglich ist - eine Kantenerkennung analog zu Abb. 5 ist daher nicht realisiert worden. Das Einfließen von Bewegungen in die Wichtigkeits-Karte - ein Prozess der im Lenken der visuellen Aufmerksamkeit des Menschen ebenfalls eine große Rolle spielt - wurde in dem vorgestellten Algorithmus ebenfalls nicht umgesetzt.

Die vorliegende Arbeit stellt also nur einen Schritt auf dem Weg zu einer vom Menschen inspirierten Erkennung dar, weist jedoch bereits viele Parallelen auf.

5 Fazit

Das Erkennen von Symbolen und Strichzeichnungen ist aus dem menschlichen Alltag nicht wegzudenken. Es ist daher eine essenzielle Fähigkeit für humanoide Roboter, wenn sie in der Lage sein sollen, mit Menschen in einer für uns natürlichen Art und Weise zu interagieren. Die in dieser Arbeit entwickelte Linienerkennung baut auf der bereits existierenden Bildvorverarbeitung Myons auf und bietet die Grundlage für diese wichtige Fähigkeit mittels eines kompakten und echtzeitfähigen Algorithmus.

Zusammenhängende Bildbereiche werden auf Basis der Häufigkeit ihrer Farbe bewertet und ihre Kanten werden vom Rest des Bildes isoliert. Durch diese Bewertung entsteht eine Wichtigkeits-Karte der vorliegenden Bildbereiche, in der nur die seltenen Farben als wichtig erachtet werden. Auf diese Weise wirken sich leichte Helligkeitsgradienten und Farbveränderungen im Hintergrund nicht auf die Linienerkennung aus, ohne dass eine Annahme über die im Bild vorkommenden Farben getroffen werden muss.

Für den Umgang mit Menschen im Alltag ist dies eine wichtige Voraussetzung, da die erkannten Farbwerte durch Veränderungen der Lichtverhältnisse stark variieren können. Aus den Kanten werden dann abstrakte Linien gebildet, welche die Grundlage für die anschließende Erkennung des Gezeichneten bieten sollen. Da bei der aus Effizienzgründen vergleichsweise einfach gehaltenen Methode zur Erkennung dieser Linien auch unerwünschte Linien als Artefakte entstehen, muss die resultierende Liste an Linien voraussichtlich bereinigt werden; Ansätze dazu wurden vorgestellt, jedoch im Rahmen dieser Arbeit nicht getestet oder realisiert. Auch die geplante Symbolerkennung auf Basis dieser Linienobjekte wurde wegen Zeitmangels noch nicht implementiert.

Während das Ziel, einen Algorithmus zur Echtzeit-Erkennung von Strichzeichnungen zu entwickeln, also nicht erreicht wurde, konnte mit der vorliegenden Arbeit ein großer Schritt zur Realisierung dieses Zieles gemacht und die bereits absehbaren verbleibenden Schritte konzeptionell vorbereitet werden.

A Glossar

In diesem Abschnitt werden Konzepte und Begriffe eingeführt, die in dieser Arbeit verwendet werden. Die Beschreibungen in diesem Glossar sollen keine allgemeingültigen Definitionen darstellen, sondern eine für das Verständnis dieser Arbeit ausreichende Erklärung liefern.

A.1 Roboter

In dieser Arbeit wird die Definition von Corke [21, Kapitel 1] verwendet: Er definiert einen Roboter als „eine ziel-orientierte Maschine, welche planen, handeln und ihre Umgebung wahrnehmen kann.“ (Übers. d. A..) Dies mag restriktiv erscheinen; nach dieser Definition gelten nur wenige Objekte, die umgangssprachlich als „Roboter“ verstanden werden, tatsächlich als Roboter.

So werden komplexe industrielle Maschinen zum Beispiel teils als „Roboter“ bezeichnet, wenngleich sie ihre Umgebung nicht wahrnehmen und auch nicht planungsfähig sind: Da sie durch die fehlende Wahrnehmung ihrer Umgebung nicht in der Lage sind, ihr Verhalten anzupassen und damit eine Gefahr für in ihrem Arbeitsbereich vorhandene Menschen darstellen können, muss meist ein klar begrenzter Bereich um solche Maschinen frei gehalten werden.

Gleichzeitig ist allein eine Wahrnehmung der Umgebung nicht ausreichend, um ein Objekt als „Roboter“ zu bezeichnen - sonst könnten bspw. Überwachungskameras „Roboter“ genannt werden.

Es gibt auch solche Maschinen, die sowohl handlungsfähig als auch in der Lage sind, ihre Umgebung wahrzunehmen - dazu zählen bspw. Staubsaugroboter, aber genau genommen auch Lampen mit Bewegungsmelder.

Diejenigen heutzutage existierenden Maschinen, welche Corkes Definition eines Roboters wahrscheinlich am meisten entsprechen, sind zum einen humanoide Roboter mancher Firmen und Labore und zum anderen Systeme, die explizit für komplexe Umgebungen entwickelt wurden (bspw. Kandidaten für „Micromouse“-Wettbewerbe). In beiden Fällen beschränkt sich die Planung meist auf Bewegungsplanung, um Terrain flexibler bewältigen zu können, oder von einem Punkt zum anderen zu gelangen.

Selbst in diesen Systemen ist die Fähigkeit des Planens also bestenfalls rudimentär vorhanden - zumindest im Vergleich zum Menschen. Auch Myon (vgl. Abs. 1.1) fehlt diese Fähigkeit zum Zeitpunkt dieser Arbeit noch. Dennoch fiel die Entscheidung hier auf diese Definition, da sie eine sehr kompakte Formulierung des perspektivisch verfolgten Ziels liefert: Der Roboter wird als Akteur in seiner Umwelt darstellt, nicht einfach als eine komplexe Maschine.

A.2 Wahrnehmung von Farben

Wird im allgemeinen Sprachgebrauch der Name einer Farbe, bspw. „Rot“, verwendet, korrespondiert das Wort „Rot“ semantisch mit einem gewissen Bereich des Farbraums. Was wir allerdings als „Rot“ bezeichnen lässt sich nicht allgemeingültig als eine Liste an Farbwerten festlegen (vgl. Abs. 1.1.3). Der Grund dafür ist, dass in der visuellen Verarbeitung des Menschen Farben kontextbedingt wahrgenommen werden und unser Verständnis von Farben entsprechend sehr viel flexibler ist, als es auf den ersten Blick erscheinen mag:

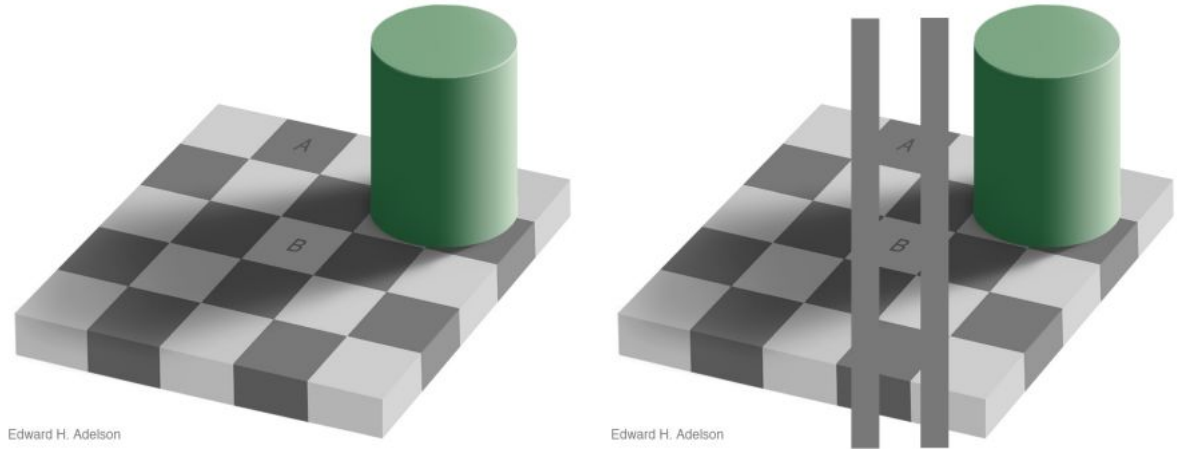


Abbildung 29: Die bekannte „Checker Shadow Illusion“: Die Quadrate „A“ und „B“ haben die gleiche Farbe, werden jedoch im linken Bild als unterschiedlich hell wahrgenommen. Das rechte Bild hilft dabei, die Gleichfarbigkeit zu erkennen, indem der Kontext verändert wird.[10]

Die in Abb. 29 dargestellte optische Illusion zeigt auf, wie sehr die Farben, die wir wahrnehmen, von den „eigentlichen“ Farben abweichen können. (Da das Konzept von Farbe inhärent subjektiv ist, lässt sich nicht einfach von einer „eigentlichen“ Farbe sprechen. Gemeint ist, dass die Farbwerte der Pixel/des Papiers, welche links als heller oder dunkler wahrgenommen werden, identisch sind.)

Dies kann in der Bildverarbeitung zu Problemen führen: Digitalkameras speichern für Pixel in einem abgelichteten Bild den „eigentlichen“ Farbwert ab. Wird das digitale Bild anschließend von einem Menschen betrachtet, so werden die Farbwerte der Pixel in der menschlichen Wahrnehmung genau so verarbeitet wie die Farbwerte des Objekts. Es liegen dennoch im digitalen Bild nicht die wahrgenommenen, sondern die „eigentlichen“ Farbwerte vor. Bei einer anschließenden Bearbeitung des Bildes durch einen (naiven) Algorithmus können durch diesen Unterschied intuitiv falsch erscheinende Ergebnisse entstehen, da unsere Intuition nicht auf den „eigentlichen“ Farbwerten arbeitet.

Als Gedankenexperiment kann angenommen werden, dass ein „roter“ Ball unter direkter Mittagssonne betrachtet und fotografiert wird, und später am Abend der selbe Ball während der Dämmerung erneut betrachtet und fotografiert wird. Für eine betrachtende Person würde die Farbe des Balls wahrscheinlich gleich erscheinen, da das menschliche Gehirn die aus den veränderten Lichtverhältnissen resultierenden Unterschiede entfernen würde - diese „Blindheit für Farbunterschiede“ tritt insbesondere dann auf, wenn die Aufmerksamkeit nicht explizit darauf gerichtet wird, die Farbe „korrekt“ wahrzunehmen.

[10] Copyright 1995, Edward H. Adelson. Bildquelle: `https://persci.mit.edu/gallery/checkershadow`

Während auch Digitalkameras meist automatisch gewisse Anpassungen der Farbwerte im Bild vornehmen, sind diese bei weitem nicht so komplex wie jene Veränderungen, welche das menschliche Gehirn unternimmt, weshalb die gespeicherten „eigentlichen“ Farbwerte des Balls in beiden Bildern nennenswert voneinander abweichen würden. Würde dieses Experiment zum Vergleich von Farben bei verschiedenen Lichtverhältnissen tatsächlich durchgeführt, so könnte außerdem auffallen, dass die „eigentlichen“ Farbwerte des Balls auch bei gleichbleibender Belichtung nicht über den gesamten Ball einheitlich sind, selbst wenn es zunächst so scheinen kann: Unter der Mittagssonne könnte es Reflexionen der Sonne auf dem Ball geben, welche „weiße“ Farbwerte hätten, darum herum gäbe es Gradienten zwischen dem „Rot“ und dem „Weiß“, et cetera.

Hinzu kommt, dass sowohl schwache Reflexionen als auch die Farbe des aktuellen Lichts immer die „eigentlichen“ Farbwerte beeinflussen. In vielen Kameras wird dies automatisch durch einen automatischen Weißabgleich und dergleichen ausgeglichen - dies ist bei Myons Kamera ebenfalls möglich, wird jedoch für diese Arbeit nicht verwendet, da die genauen Farbwerte weniger relevant sind als die relativen Unterschiede der Farbwerte zueinander. Es kann dadurch in manchen Bildern zu farblichen Abweichungen kommen.

A.3 Morphologische Operationen

Bei der folgenden Beschreibung von morphologischen Operationen handelt es sich um eine Zusammenfassung von Corke [21, Kapitel 12.5]; für eine formale Definition und eine umfassende Behandlung wird auf diese Quelle verwiesen. Das Ziel dieses Abschnitts ist keine formale Definition, sondern ein grundlegendes Verständnis dafür, was die Operationen im Bild bewirken.

Als Minkowski Subtraktion, oder auch Erosion, versteht man eine Operation in der Bildverarbeitung, welche auf binäre Bilder angewandt wird. Dafür wird zunächst eine Matrix als sog. Strukturelement (SE) definiert, welches anschließend auf das Bild angewandt wird; das Ergebnis ist ein neues Bild gleicher Größe. Ähnlich wie bei der Faltung gibt es verschiedene Möglichkeiten, mit dem Rand des Bildes umzugehen, die unterschiedliche Ergebnisse liefern können. In dieser Arbeit wird um das zu bearbeitende Bild ein Rahmen aus „0“ Werten angenommen, was dazu führt, dass die Werte der äußersten Pixel des resultierenden Bildes „0“ sind.

Um ein Verständnis der Operation zu erhalten, kann man sich vorstellen, dass das gewählte Strukturelement schrittweise über das zu bearbeitende Bild gelegt wird, so dass das Zentrum des SE jeweils einmal auf jedem Pixel des Bildes liegt. In das resultierende neue Bild wird eine „1“ für das aktuell betrachtete Pixel eingetragen, wenn „unter“ *jeder* „1“ im SE auch eine „1“ im zu bearbeitenden Bild vorkommt - andernfalls wird eine „0“ ins neue Bild eingetragen; es besteht also eine Parallele zu dem logischen Und ($\wedge$).

Die spezifische Form des SE ist entscheidend dafür, welche Formen im resultierenden Bild erhalten bleiben. Ein beispielhaftes SE $\mathbf{S}$, mit dem nach einer Erosion vertikale Linien mit einer Mindestlänge von drei Pixeln im Bild erhalten blieben, während alles andere entfernt würde, ist hier dargestellt:

$$\mathbf{S} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Dabei ist zu beachten, dass die vertikalen Linien im resultierenden Bild außerdem um jeweils ein Pixel oben und unten verkürzt vorliegen würden (vgl. Abb. 31).

Der übliche Operator für die Minkowski Subtraktion ist $\ominus$, also

$$\mathbf{I}' = \mathbf{I} \ominus \mathbf{S}, \tag{1}$$

wobei $\mathbf{I}$ das originale Bild, $\mathbf{S}$ das Strukturelement und $\mathbf{I}'$ das Ergebnis der Operation bezeichnen. Eine Visualisierung der Minkowski Subtraktion mit dem genannten Strukturelement ist in Abb. 31 gegeben, während Abb. 30 das Ergebnis zeigt:

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&1&0&0&1&0&0&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&1&1&1&1&1&1&1&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \ominus \begin{bmatrix} 0&1&0\\ 0&1&0\\ 0&1&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&1&0&0&1&0&0&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix}$$

Abbildung 30: Darstellung der Minkowski Subtraktion: Links ist das originale Bild ($\mathbf{I}$ in Gl. (1)), mittig ist das Strukturelement ($\mathbf{S}$) und rechts das Ergebnis der Operation ($\mathbf{I}'$) abgebildet.

$$\begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&1&0&0&1&0&0&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&1&1&1&1&1&1&1&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&0 \end{bmatrix} \ominus \begin{bmatrix} 0&1&0\\ 0&1&0\\ 0&1&0 \end{bmatrix} = \begin{bmatrix} 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0\\ ?&1&?&?&?&?&?&0&?\\ ?&?&?&?&?&?&?&?&?\\ ?&?&?&?&?&?&?&?&?\\ ?&?&?&?&?&?&?&?&?\\ ?&?&?&?&?&?&?&?&?\\ ?&?&?&?&?&?&?&?&? \end{bmatrix}$$

Abbildung 31: Darstellung des Prozesses der Minkowski Subtraktion. Die Minkowski Subtraktion wurde nur zum Teil durchgeführt, um die Gelegenheit zu haben, die Operation zu verdeutlichen: Es wurden zwei Pixel im originalen Bild (links, entsprechend $\mathbf{I}$ aus Gl. (1)) ausgewählt und deren „Bewertung“ durch das Strukturelement (mittig, $\mathbf{S}$) dargestellt, wobei Blau eine Übereinstimmung zwischen $\mathbf{S}$ und $\mathbf{I}$ und Rot eine Diskrepanz zwischen beiden anzeigen soll. Da in dem rot umrandeten Bereich eine Diskrepanz auftritt, wird dem entsprechenden Pixel im resultierenden Bild (rechts, $\mathbf{I}'$) der Wert „0“ zugewiesen, wie zuvor erklärt. Da in dem blau umrandeten Bereich keine Diskrepanz auftritt, wird rechts entsprechend eine „1“ eingetragen. (Nur Diskrepanzen mit dem Bereich, der im Strukturelement mit Einsen gefüllt ist, sind hier relevant. Irrelevante Pixel im originalen Bild wurden, um dies zu verdeutlichen, mit einem mittleren Grau hinterlegt.)

Die entsprechende additive Operation heißt Minkowski Addition (oder Dilatation) und wird mit dem Operator $\oplus$ gekennzeichnet. Dabei wird in das neue Bild eine „1“ eingetragen, wenn „unter“ *einer* „1“ im SE auch eine „1“ im zu bearbeitenden Bild vorkommt; das Verhalten ist also analog zum logischen Oder ($\wedge$).

Ähnlich wie bei den assoziierten logischen Operationen gilt: Die Dilatation ist *keine* inverse Operation zur Erosion.

$$\mathbf{I} \neq (\mathbf{I} \ominus \mathbf{S}) \oplus \mathbf{S}$$

Stattdessen würde dies ein sog. „morphologisches Öffnen“ zur Folge haben: Die nach der Erosion noch vorliegenden Objekte werden durch die Dilatation wiederhergestellt, doch alle Objekte, die durch die Erosion verloren gingen, können nicht zurückgewonnen werden; ein Beispiel dafür ist in Abb. 32 gegeben. Die Wiederherstellung erfolgt ggf. nur partiell, in Abhängigkeit des gewählten SE. Der zugehörige Operator ist $\circ$: $\mathbf{I} \circ \mathbf{S} = (\mathbf{I} \ominus \mathbf{S}) \oplus \mathbf{S}$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&1&0&0&0&0\\
\hline
0&1&0&0&1&0&0&0&0\\
\hline
0&1&0&0&1&0&0&1&0\\
\hline
0&0&0&0&1&0&0&0&0\\
\hline
0&0&0&0&1&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0\\
\hline
\end{array}
\oplus
\begin{array}{|c|c|c|}
\hline
0&1&0\\
\hline
0&1&0\\
\hline
0&1&0\\
\hline
\end{array}
=
\begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&1&0&0&0&0\\
\hline
0&1&0&0&1&0&0&0&0\\
\hline
0&1&0&0&1&0&0&1&0\\
\hline
0&1&0&0&1&0&0&1&0\\
\hline
0&1&0&0&1&0&0&1&0\\
\hline
0&0&0&0&1&0&0&0&0\\
\hline
0&0&0&0&1&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0\\
\hline
\end{array}$$

Abbildung 32: Darstellung des morphologischen Öffnens: Links ist das Ergebnis aus Abb. 30 ($\mathbf{I} \ominus \mathbf{S}$), mittig ist das Strukturelement ($\mathbf{S}$) und rechts das Ergebnis der Operation ($(\mathbf{I} \ominus \mathbf{S}) \oplus \mathbf{S}$) abgebildet. Es ist eindeutig nicht das Original aus Abb. 30, da die horizontale Linie nicht wiederhergestellt werden konnte.

Die verschiedenen möglichen Kombinationen von Erosion und Dilatation werden gesammelt als „morphologische Operationen“ bezeichnet.

A.4 Nachbarschaft

Sowohl im Kontext der Zellen einer Matrix als auch im Kontext biologischer Zellen wird in dieser Arbeit der Begriff der „Nachbarschaft“ verwendet. In beiden Fällen werden Zellen dann als „benachbart“ verstanden, wenn ihre Chebyshev-Distanz zueinander „1“ beträgt.

Im Falle einer Matrix bedeutet dies, dass die jeweilige Differenz der Indizes der betrachteten Zellen entlang jeder Dimension maximal „1“ sein darf - stellt man sich eine zweidimensionale Matrix als Gittermuster vor, so sind alle Zellen benachbart, welche mindestens eine gemeinsame Ecke besitzen. Im Falle biologischer Zellen ist mit „Nachbarschaft“ gemeint, dass zwischen zwei Zellen keine andere Zelle des gleichen Typs Platz hätte.

Eine Zelle ist sich selbst kein Nachbar.

A.5 Neuronales Netz

Ein künstliches Neuronales Netz (NN) besteht aus künstlichen Neuronen.[11] Als künstliches Neuron bezeichnet man eine mathematische Funktion, welche die Funktionalität biologischer Neuronen modellieren soll.

Jedes künstliche Neuron kann beliebig viele Eingangssignale erhalten, hat jedoch nur ein Ausgangssignal. Dieses Ausgangssignal wird mittels der sog. Aktivierungsfunktion aus den Eingangssignalen berechnet. Zu jedem künstlichen Neuron gehört neben der Aktivierungsfunktion außerdem ein Vektor, welcher die „Gewichte“ des künstlichen Neurons beinhaltet. Bei diesen Gewichten handelt es sich um Faktoren, welche vor der Aktivierungsfunktion auf die Eingangssignale angewandt werden; die Gewichte können unabhängig voneinander verändert werden, um den Einfluss spezifischer Eingangssignalquellen auf die Aktivierung des künstlichen Neurons zu modulieren.

NNs bestehen aus mindestens zwei künstlichen Neuronen, deren Vernetzung von der Anwendung abhängt. Häufig sind NNs in sog. „Schichten“ (engl. *layers*) aufgebaut: Bei einem solchen Aufbau bilden jeweils mehrere künstliche Neuronen gemeinsam eine Schicht des NNs und jedes künstliche Neuron einer Schicht erhält üblicherweise die Ausgangssignale der künstlichen Neuronen der vorherigen Schicht als Eingangssignale. Meist werden die Schichten solcher NNs in einer festen Reihenfolge aktiv, ohne dass es zu einer Rückführung von Signalen kommt: Die erste Schicht erhält ein Eingangssignal (bspw. ein Bild), woraufhin die resultierende Aktivierung Schicht für Schicht durch das gesamte NN propagiert, bis am Ende die letzte Schicht das Ausgangssignal des gesamten NNs liefert - alle Schichten zwischen der ersten und der letzten werden i.d.R. nicht ausgelesen; man spricht von „versteckten Schichten“ (engl. *hidden layers*).

Dank dieses geschichteten Aufbaus kann die Veränderung der Gewichte einzelner Neuronen starken Einfluss auf die Ausgangssignale aller Neuronen in folgenden Schichten haben.

[11] Diese Erklärung von NNs und künstlichen Neuronen ist eine sehr vereinfachte und generalisierte Darstellung.

B Erweiternde Abbildungen

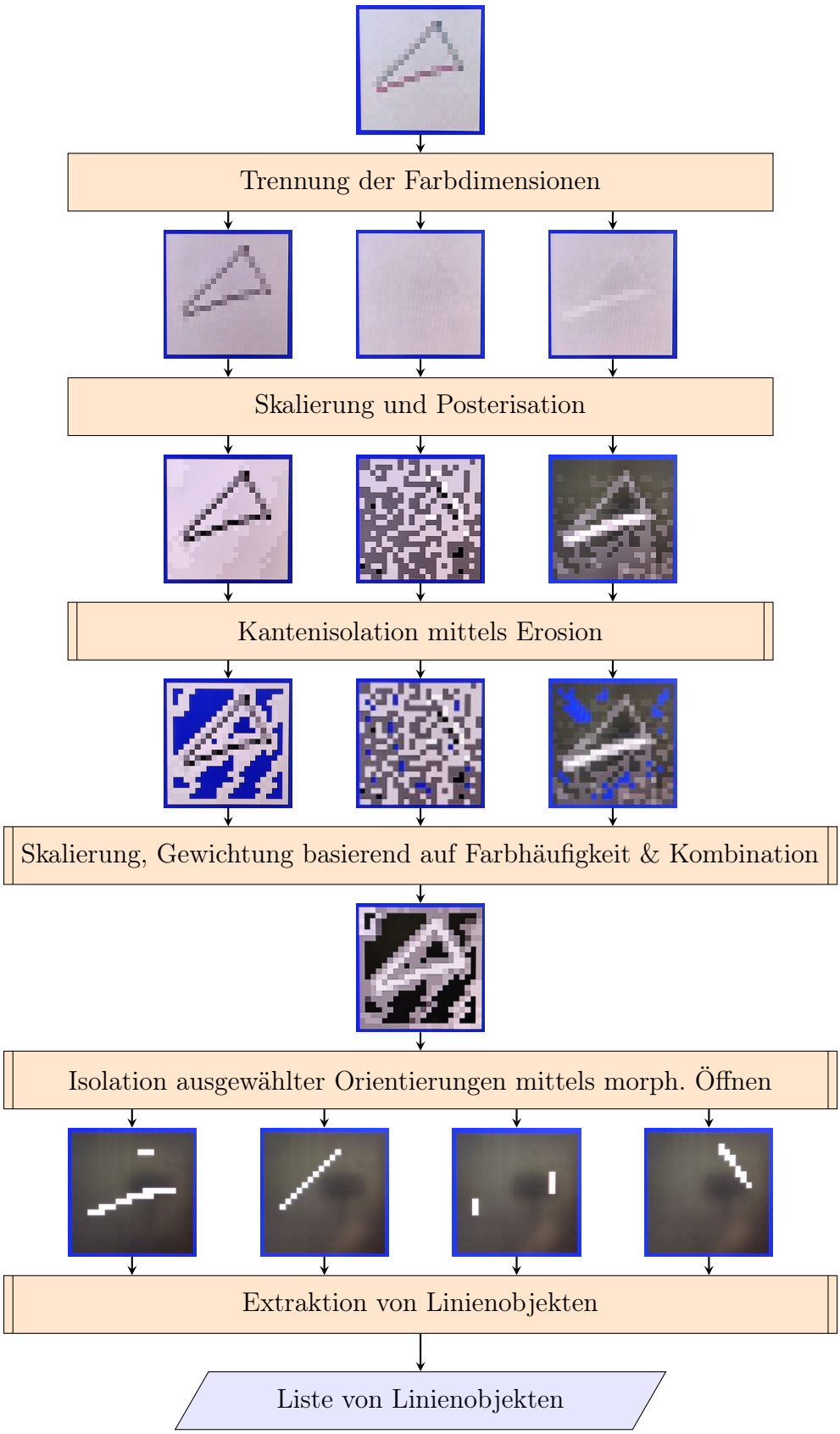


Abbildung 33: Variante des Datenflussdiagramms (Abb. 6) aus Abs. 3.1 mit bildlicher Darstellung einzelner Zwischenergebnisse.

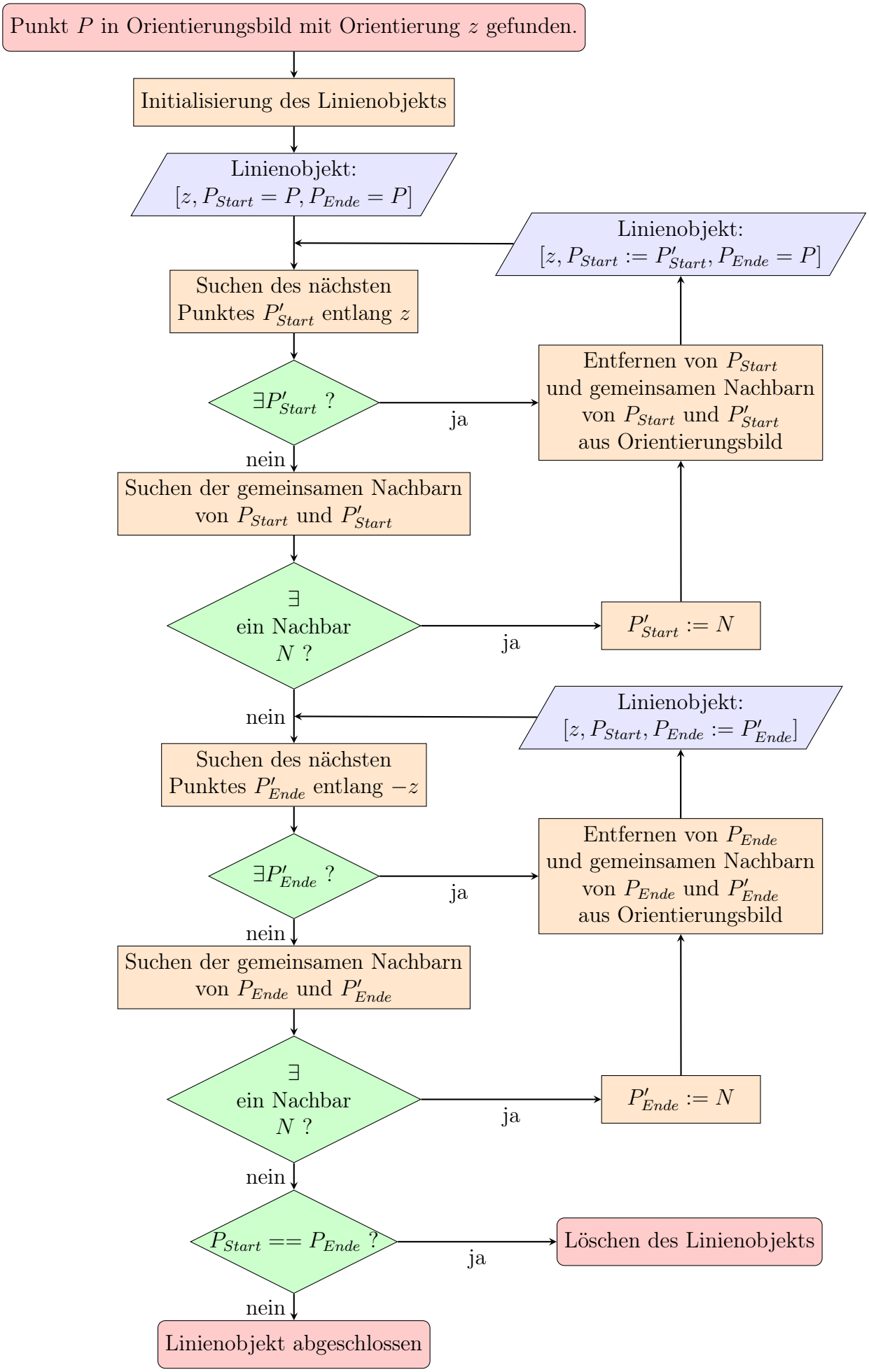
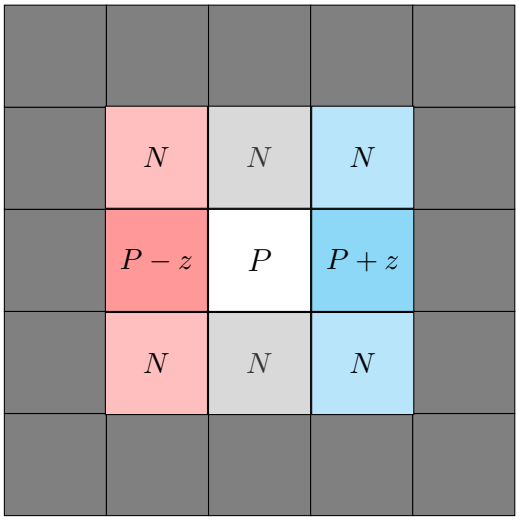
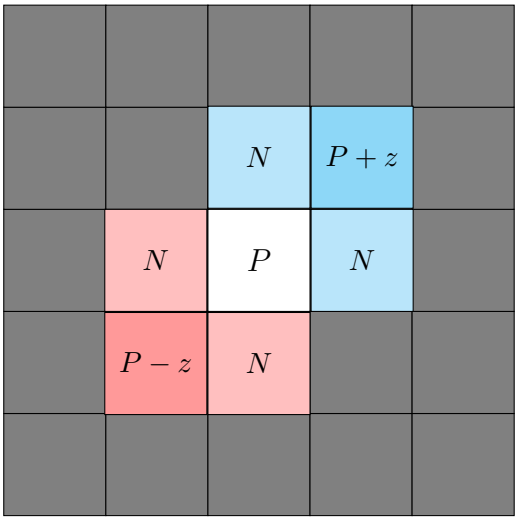


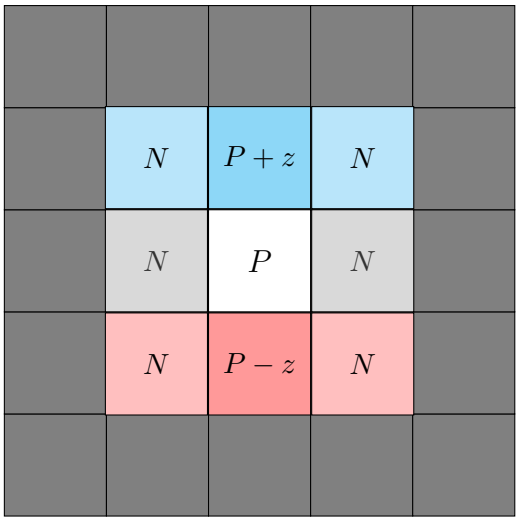
Abbildung 34: Erweiterung des Prozesses „Extraktion von Linienobjekten“ aus Abb. 6 (und Abb. 33). Für eine Erklärung von „gemeinsamen Nachbarn“ und dem „nächsten Punkt entlang z (bzw. $-z$)“ siehe Abb. 35.



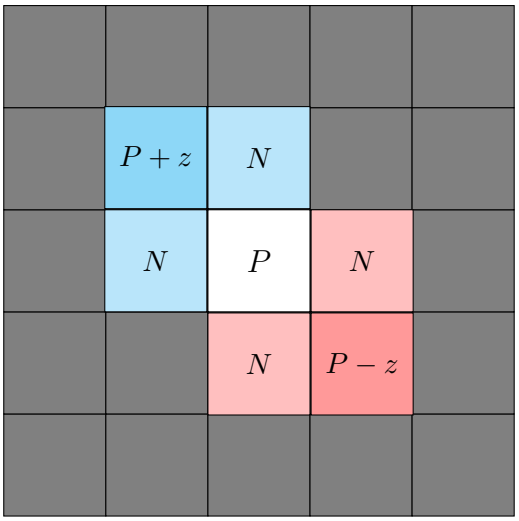
Orientierung $z = 0°$



Orientierung $z = 45°$



Orientierung $z = 90°$



Orientierung $z = 135°$

Abbildung 35: Visualisierung der „gemeinsamen Nachbarn“ und „nächsten Punkte entlang z“ aus Abb. 34. Der betrachtete Punkt ist die mit P bezeichnete Zelle. Relativ dazu ist $P + z$ der nächste Punkt entlang z und $P - z$ entsprechend der nächste Punkt entlang $-z$. Die mit N bezeichneten Zellen sind Nachbarn von P (vgl. App. A.4) und die Füllfarbe der Zellen gibt an, ob sie außerdem zu $P + z$ oder $P - z$ benachbart sind; die mit hellgrau gefüllten Zellen sind mit $P + z$ *und* $P - z$ benachbart. Die „gemeinsamen Nachbarn“ zweier Punkte sind also die Teilmenge der Nachbarn dieser Punkte: Zwischen P und $P \pm z$ gibt es bei einer Orientierung von 0° oder 90° jeweils vier gemeinsame Nachbarn, bei einer Orientierung von 45° oder 135° jedoch nur jeweils zwei. Für den in Abb. 34 dargestellten Algorithmus werden die grau hinterlegten Nachbarn jedoch *nicht* betrachtet, da sie zuvor bereits überprüft wurden - die einzige Ausnahme bildet der erste Schritt nach der Initialisierung; diese Ausnahme wird vernachlässigt. Die frühere Prüfung erfolgte in einem früheren Schritt der Rekursion, als der jetzige Punkt P noch als Punkt $P + z$ oder $P - z$ betrachtet wurde.

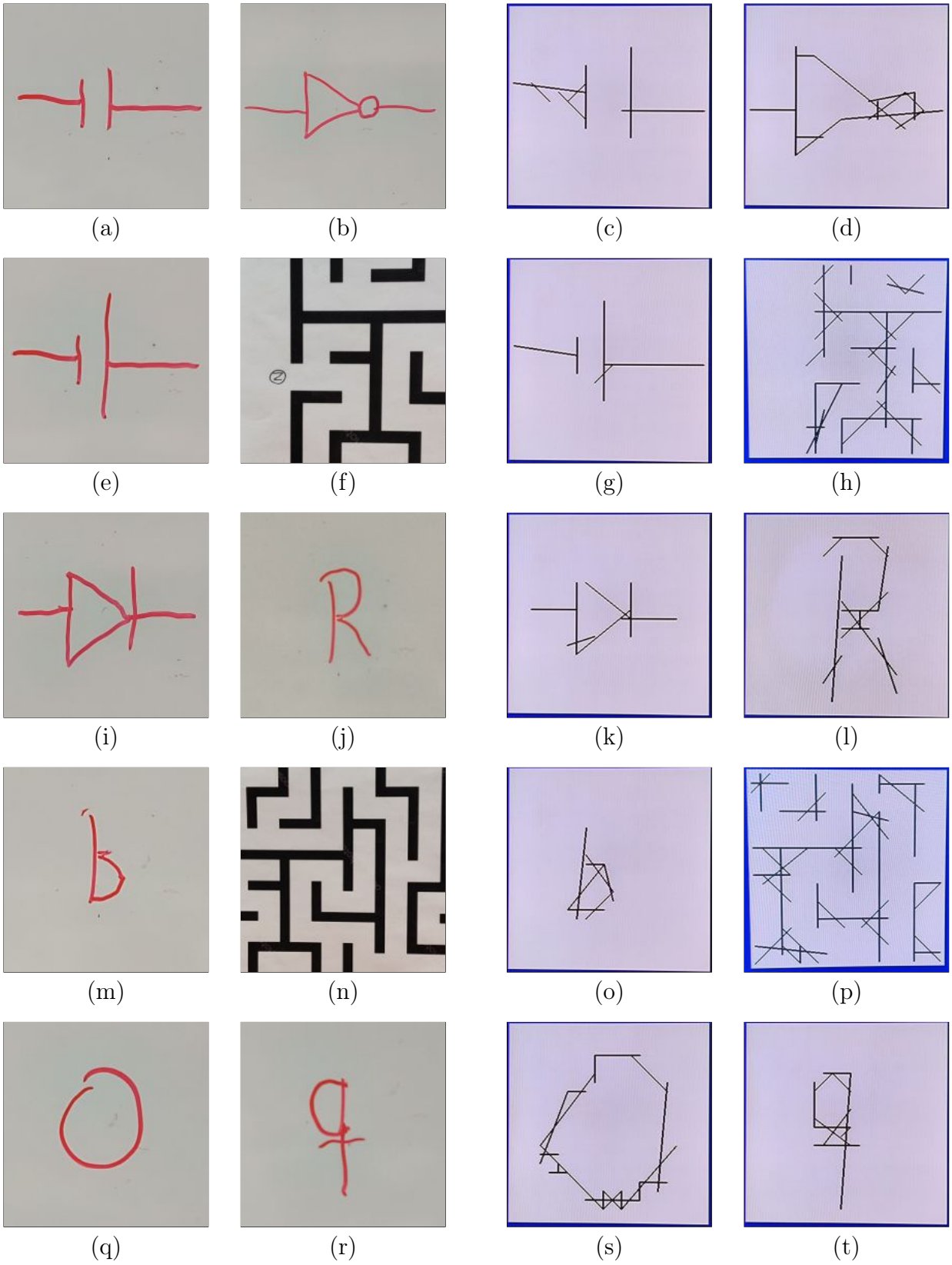


Abbildung 36: Alternative, günstigere Re-Synthesen zu manchen der gezielt ungünstig gewählten Bilder aus Abb. 25 und 26.

Literatur

[1] D. Man und R. Olchawa, „The Possibilities of Using BCI Technology in Biomedical Engineering," in Feb. 2018, S. 30–37, ISBN: 978-3-319-75024-8. DOI: 10.1007/978-3-319-75025-5_4.

[2] E. Yudkowsky, *The Lens That Sees Its Flaws*, 2007. Adresse: https://www.lesswrong.com/posts/46qnWRSR7L2eyNbMA/the-lens-that-sees-its-flaws (besucht am 17. 11. 2024).

[3] A. Benoit, A. Caplier, B. Durette und J. Herault, „Using Human Visual System modeling for bio-inspired low level image processing," *Computer Vision and Image Understanding*, Jg. 114, Nr. 7, S. 758–773, 2010, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2010.01.011. Adresse: https://www.sciencedirect.com/science/article/pii/S1077314210000482.

[4] J. Krauskopf, „Chapter 3 - Color Vision," in *Color for Science, Art and Technology*, Ser. AZimuth, K. Nassau, Hrsg., Bd. 1, North-Holland, 1998, S. 97–121. DOI: https://doi.org/10.1016/S1387-6783(98)80006-8. Adresse: https://www.sciencedirect.com/science/article/pii/S1387678398800068.

[5] L. Zhaoping, „A new framework for understanding vision from the perspective of the primary visual cortex," *Curr Opin Neurobiol. 2019 Oct;58:1-10*, 2019. DOI: 10.1016/j.conb.2019.06.001.

[6] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov und S. Carter, „Zoom In: An Introduction to Circuits," *Distill*, 2020, https://distill.pub/2020/circuits/zoom-in. DOI: 10.23915/distill.00024.001.

[7] L. Zhaoping und K. A. May, „Psychophysical tests of the hypothesis of a bottom-up saliency map in primary visual cortex," en, *PLoS Comput. Biol.*, Jg. 3, Nr. 4, e62, Apr. 2007.

[8] S. Wend, *Entwicklung und vergleichende Bewertung eines Algorithmus zur Erkennung von Strichzeichnungen*, Juli 2024.

[9] B. Panreck, „Entwicklung einer heuristischen Verhaltensregelung für die Visuomotorik humanoider Roboter," Diss., Freie Universität Berlin, 2021.

[10] J. Lehman, J. Clune, D. Misevic u. a., *The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities*, 2019. arXiv: 1803.03453 [cs.NE]. Adresse: https://arxiv.org/abs/1803.03453.

[11] Y.-Q. Cheng und J. Yang, „Effective recognition approach to assembly drawings," in *Applications of Digital Image Processing XIII*, A. G. Tescher, Hrsg., International Society for Optics und Photonics, Bd. 1349, SPIE, 1990, S. 396–404. DOI: 10.1117/12.23555. Adresse: https://doi.org/10.1117/12.23555.

[12] E. L'Homer, „Extraction of strokes in handwritten characters," *Pattern Recognit.*, Jg. 33, S. 1147–1160, 2000. DOI: 10.1016/s0031-3203(99)00103-x. Adresse: https://api.semanticscholar.org/CorpusID:15378655.

[13] M. R. Wróbel, J. T. Starczewski, K. Nieszporek, P. Opielka und A. Kazmierczak, „A Greedy Algorithm for Extraction of Handwritten Strokes," in *International Conference on Artificial Intelligence and Soft Computing*, 2019. Adresse: https://api.semanticscholar.org/CorpusID:167222004.

[14] K. L. Bauer und W. A. Barrett, „Intelligent Pen: A Least Cost Search Approach to Stroke Extraction in Historical Documents,“ in *Document Recognition and Retrieval*, 2016. Adresse: https://api.semanticscholar.org/CorpusID:38878314.

[15] W. Lee, L. Kara und T. Stahovich, „An efficient graph-based recognizer for hand-drawn symbols,“ *Computers & Graphics*, Jg. 31, S. 554–567, Aug. 2007. DOI: 10.1016/j.cag.2007.04.007.

[16] A. Bhattacharya, S. Roy, N. Sarkar, S. Malakar und R. Sarkar, „Circuit Component Detection in Offline Handdrawn Electrical/Electronic Circuit Diagram,“ in *2020 IEEE Calcutta Conference (CALCON)*, Feb. 2020, S. 80–84. DOI: 10.1109/CALCON49167.2020.9106527.

[17] D. Dörner, *Bauplan für eine Seele*. Jan. 2001.

[18] N. Otsu, „A Threshold Selection Method from Gray-Level Histograms,“ *IEEE Transactions on Systems, Man, and Cybernetics*, Jg. 9, Nr. 1, S. 62–66, Jan. 1979, ISSN: 2168-2909. DOI: 10.1109/TSMC.1979.4310076.

[19] J. Sauvola und M. Pietikäinen, „Adaptive document image binarization,“ *Pattern Recognition*, Jg. 33, Nr. 2, S. 225–236, 2000, ISSN: 0031-3203. DOI: https://doi.org/10.1016/S0031-3203(99)00055-2. Adresse: https://www.sciencedirect.com/science/article/pii/S0031320399000552.

[20] G. Cohen, S. Afshar, J. Tapson und A. van Schaik, *EMNIST: an extension of MNIST to handwritten letters*. 2017. Adresse: http://arxiv.org/abs/1702.05373.

[21] P. Corke, *Robotics, Vision and Control*. Springer Berlin, Heidelberg, 2011. DOI: https://doi.org/10.1007/978-3-642-20144-8.

Abbildungsverzeichnis

1 Der humanoide Roboter Myon . . . 2
2 Kombination von Pixeln zu Superpixeln . . . 3
3 Visualisierung der UV-Ebene bei konstantem Y-Wert . . . 4
4 Darstellung der Nervenbahnen von den Augen zum primären visuellen Kortex 7
5 Schematische Darstellung der Wahrnehmung einer „Kante“ zwischen zwei „Flächen“ unterschiedlich orientierter Linien durch den V1 . . . 9
6 Datenflussdiagramm des entwickelten Algorithmus . . . 16
7 Arbeitsbeispiel für den vorgestellten Algorithmus . . . 17
8 Darstellung der einzelnen Farbkanäle des Bildes . . . 18
9 Ergebnis der Skalierung und Posterisation der Bilder aus Abb. 8 . . . 18
10 Beispiel für Erosionsvariante mit Gleichheit . . . 19
11 Prozess der Kantenisolation durch Erosion am Beispiel des Y-Kanals . . . 19
12 Aus der Umgewichtung resultierende Falschfarbenbilder der zuvor isolierten Bildkanten . . . 20
13 Darstellung der Wichtigkeits-Karte vor und nach der Anwendung des Schwellwerts . . . 21
14 Isolation der Orientierungen mittels morphologischem Öffnen . . . 22
15 Darstellung der gefundenen Linienobjekte . . . 23
16 Vergleich mit L’Homers Stroke Extraction . . . 27
17 Zwischenergebnisse im Erkennen von MNIST-Bildern . . . 28
18 Weiterer Vergleich mit L’Homers Stroke Extraction . . . 29
19 Linienerkennung an einem einfach gezeichneten Haus auf weißem Papier . . . 30
20 Linienerkennung für Tic-Tac-Toe auf weißem Papier . . . 30
21 Linienerkennung für einfache Symbole auf weißem Papier . . . 31
22 Linienerkennung auf kariertem Papier . . . 32
23 Linienerkennung bei einem simplen, unbeschrifteten Schaltplan auf kariertem Papier . . . 33
24 Linienerkennung bei einem Schaltplan auf einem Whiteboard . . . 34
25 Beispiel für die Erkennbarkeit von re-synthetisierten Strichzeichnungen . . . 36
26 Weiteres Beispiel für die Erkennbarkeit von re-synthetisierten Strichzeichnungen 37
27 Darstellung von Variabilität in der Linienerkennung . . . 38
28 Laufzeitverhalten des vorgestellten Algorithmus . . . 39
29 Die bekannte „Checker Shadow Illusion“ . . . 43
30 Darstellung der Minkowski Subtraktion . . . 45
31 Darstellung des Prozesses der Minkowski Subtraktion . . . 45
32 Darstellung des morphologischen Öffnens . . . 46
33 Variante des Datenflussdiagramms (Abb. 6) aus Abs. 3.1 mit bildlicher Darstellung einzelner Zwischenergebnisse . . . 48
34 Erweiterung des Prozesses „Extraktion von Linienobjekten“ aus Abb. 6 (und Abb. 33) . . . 49
35 Visualisierung der „gemeinsamen Nachbarn“ und „nächsten Punkte entlang z“ aus Abb. 34 . . . 50
36 Alternative, günstigere Re-Synthesen zu manchen der gezielt ungünstig gewählten Bilder aus Abb. 25 und 26 . . . 51