Institut für parallele und verteilte Systeme
Abteilung Bildverstehen
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Fachstudie Nr.

Fachstudie über Algorithmen zur Erkennung von Objekten aus CT-Voxel-Datensätzen

Marcus Hörger, Andreas Paul, Felix Reeh

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr.-Ing. Sven Simon
Betreuer:	Dipl.-Inf. Jürgen Hillebrandt, Dipl.-Inf. Steffen Kieß
begonnen am:	06. Dezember 2011
beendet am:	06. Juni 2012
CR-Klassifikation:	I.4, I.5.1

Kurzfassung

Für die computergestützte Analyse planarer Hochfrequenzschaltungen wird mit 3D-Modellen gearbeitet anhand derer die elektrischen Eigenschaften simuliert werden können. Diese 3D-Modelle werden auf Basis von Bildern, die mit Computertomographen erstellt wurden, angefertigt.
Dieser Prozess wurde bisher meistens mühsam von Hand durchgeführt. Deshalb werden Algorithmen gesucht, die dabei helfen einfache Strukturen wie Leiterbahnen, Bonddrähte, Lötkugeln und Bohrungen automatisch zu erkennen. Dazu werden im folgenden einige potentielle algorithmische Ansätze vorgestellt und bewertet.
Zu Beginn der Studie werden einige Hilfsverfahren (z.B. Canny-Edge-Detektor) vorgestellt, die anschließend an verschiedenen Stellen eingesetzt werden. Anschließend untersuchen die Autoren verschiedene Möglichkeiten zweidimensionale „Slices“ aus den dreidimensionalen Voxeldatensätzen zu extrahieren. Das Ziel dabei ist es, möglichst aussagekräftige Bilder für den Einsatz der, im folgenden vorgestellten, zweidimensionalen Objekterkennungsverfahren zu finden. Es stellt sich heraus, dass verschiedene zweidimensionale Verfahren, aufgrund der Struktur des zu untersuchenden Datensatzes, für Bohrungen und Leiterbahnen gut geeignet sind und größtenteils brauchbare Resultate liefern.
Für andere Objekte, wie Kugeln und Bonddrähte, wird der nichtdeterministische RANSAC-Algorithmus und ein auf das Problem zugeschnittener, selbstgewählter Ansatz untersucht. Im letzten Abschnitt werden nun die betrachteten Verfahren einander gegenübergestellt und anschließend eine Empfehlung bezüglich des Einsatzes in der Praxis geggeben.

Inhaltsverzeichnis

1 Einleitung

Das Forschungsthema des Aufgabenstellers befasst sich mit der computergestützten Analyse von aufgebauten, passiven und planaren Hochfrequenzschaltungen. Dies geschieht mit Hilfe von höchst präzisen 3D Modellen, die auf Basis von Messungen mit einem Computertomographen erstellt werden.
Da die Ausgaben des digitalen Röntgendetektors in Form von diskreten Voxeldaten vorliegen, werden von den derzeit eingesetzten Algorithmen zur Objekterzeugung nur Objekte mit rauen und kantigen Oberflächen erzeugt. Leider führt dies bei der Analyse zu Nachteilen bei der Bestimmung der elektrischen Hochfrequenzeigenschaften.
Um diesen Nachteil zu umgehen werden die Modelle zur Zeit größenteils von Hand erstellt. Leider stellt die manuelle Konstruktion einen nicht unerheblichen zusätzlichen Zeitaufwand dar. Deshalb wären Algorithmen wünschenswert, die die relevanten Bauteile einer solchen Schaltung weitestgehend autonom erkennen und abstrahieren können.

2 Problemstellung

Eine Möglichkeit der Verfälschung der erkannten Objekte zu begegnen ist die Ersetzung bzw. Approximierung dieser durch einfachere Objekte (z.B. Zylinder, Kugel oder Quader). Im Rahmen der Fachstudie sollten deshalb Algorithmen ausfindig gemacht werden mit denen ausgewählte Objekte innerhalb der Bilddaten erkannt werden können. Anhand einfacher Teststrukturen sollte eine Auswahl vielversprechender Algorithmen getestet und miteinander verglichen werden. Die Ergebnisse wurden protokolliert und bewertet.

3 Beschreibung der zu untersuchenden Daten

Die Ergebnisse der Messungen werden in zwei verschiedenen Formaten untersucht:

1. **3D-Voxeldaten**
   In ihrer ursprünglichen Form liegen die Daten in Form von dreidimensionalen Rastergrafiken, zusammengesetzt aus volumetrischen Pixeln, vor. Diese Pixel werden üblicherweise als Voxel bezeichnet und sind Teil eines Voxelgitters. Jeder Voxel besitzt einen Wert, der seine Opazität wiedergibt. Der Wert dieser Opazität leitet sich direkt aus der beim CT gemessenen Reflektanzwert ab.

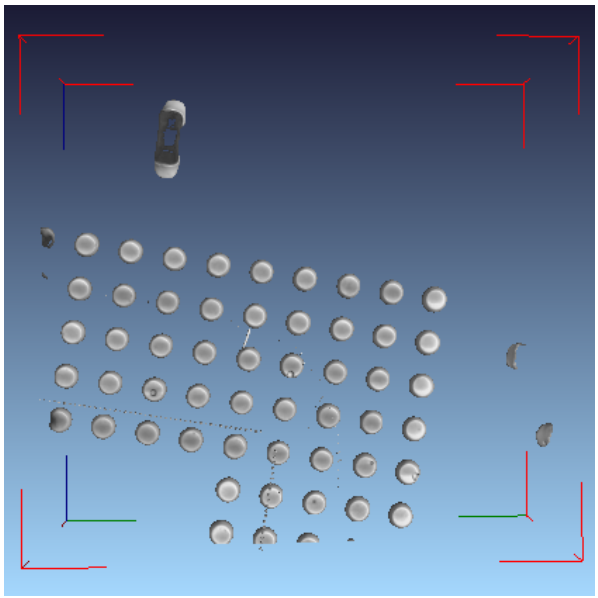

Abbildung 3.1: 3D-Voxeldatensatz in myVGL

2. **2D-Slices**
   Die zweidimensionalen Slices werden auf Basis der 3D-Voxeldaten berechnet. Dabei wird eine Schicht des Voxelgitters extrahiert und auf Basis der Opazitätswerte eine Rastergrafik erstellt, wobei jeder Pixel eindeutig einem Voxel zugeordnet werden kann. Durch diese Umformung können bei der Objekterkennung auch 2D-Verfahren genutzt werden.

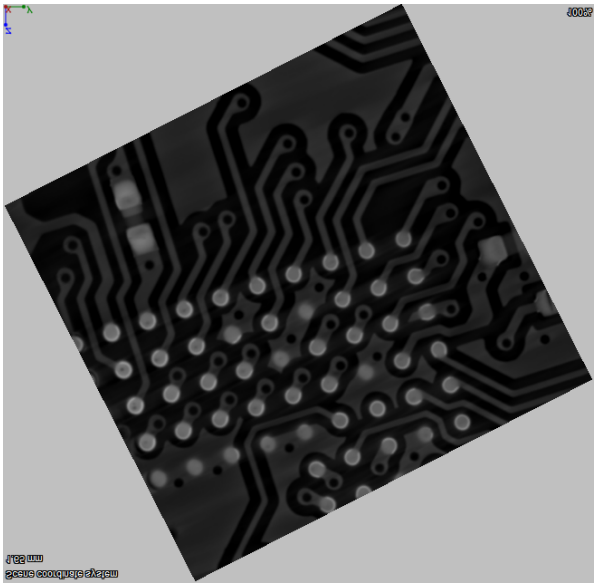

Abbildung 3.2: 2D-Slice aus einem 3D-Voxeldatensatz

3.1 Zu erkennende Objekte

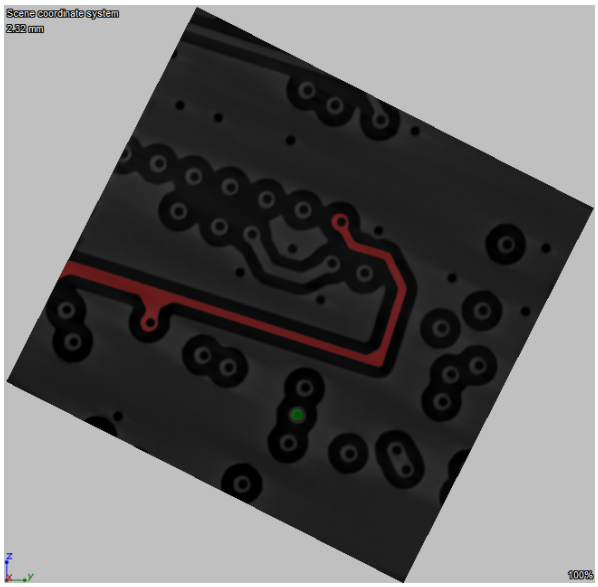

Abbildung 3.3: Typischer 2D-Slice mit Leiterbahnen (rot) und Bohrpunkten (grün)

3.1.1 Bohrpunkte

Die Bohrpunkte sind die wohl am einfachsten zu erkennenden Objekte. Ihre Größe ist meistens durchweg identisch. Häufig münden Leiterbahnen in Bohrpunkten, manchmal befinden sie sich auch dazwischen, entlang des Körpers der Leiterbahn. Für die Erkennung bietet es sich an auf den zweidimensionalen Slices zu arbeiten, weil die Bohrpunkte dort als schwarze, ausgefüllte Kreise leichter auszumachen sind.

3.1.2 Leiterbahnen

Die Leiterbahnen sind elektrisch leitende Verbindungen mit zweidimensionalem Verlauf. Aufgrund diesr Eigenschaft bietet es sich an wie bei den Bohrpunkten auf zweidimensionale Erkennungsverfahren zurückzugreifen. Da sie in den allermeisten Fällen an einem Bohrpunkt starten und auch wieder enden können diese gegebenenfalls auch als Ausgangspunkte für die Erkennung genutzt werden.

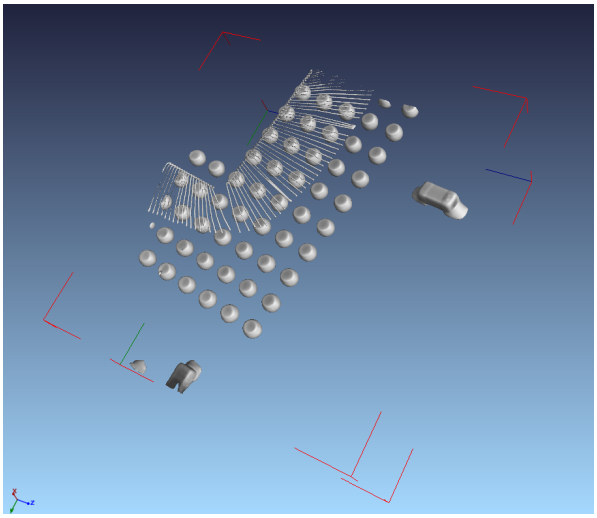

Abbildung 3.4: In der Mitte zu sehen: Bonddrähte und Lötkugeln

3.1.3 Lötkugeln

Die Lötkugeln sind die Objekte, die sich durch ihren Reflektanzwert am deutlichsten von anderen Objekten abheben. Deshalb können sie durch Filterung mit Threshholds sehr einfach vorisoliert werden. Die Herausforderung besteht demnach darin die isolierten Voxel-Mengen auf Kugel-Eigenschaften zu untersuchen und gegebenenfalls als Lötkugel zu klassifizieren. Es sind allerdings auch Verfahren im zweidimensionalen denkbar, da die Kugelen auch auf den Slices sehr gut sichtbar sind. Dann wäre es allerdings nötig Schicht für Schicht, also Slice für Slice, vorzugehen und den Vorteil eines zusammenhängenden Objektes zu verwerfen.

3.1.4 Bonddrähte

Die Bonddrähte verbinden die Schichten der Platine, sie sind dünn und lang. Sie zeigen nicht unbedingt alle in die gleiche Richtung und weisen auch alle eine unterschiedliche Länge auf. Da sie im zweidimensionalen durch ihre geringe Größe und Erstreckung über mehrere Slices kaum erkennbar sind, müssen dreidimensinale Verfahren eingesetzt werden. Durch ihre undankbaren Eigenschaften sind sie die am schwersten zu erkennenden Objekte.

4 Allgemeine Verfahren

Die Allgemeinen Verfahren sind unterstützende Algorithmen mit denen die zu analysierenden Daten für komplexere Erkennungsverfahren vorverarbeitet werden. Meistens filtern sie unnötige oder störende Informationen heraus oder modifizieren die Daten so, dass sich die zu erkennenden Objekte besser von den unbrauchbaren Daten abheben.

4.0.5 Canny-Edge-Detektor

Der Canny-Edge Operator [Can86] is ein von John Canny 1986 entwickelter Algorithmus zur Kantendetektion. Er liefert für ein Grauwertbild idealerweise alle zusammenhängenden Kanten. Der Algorithmus gliedert sich dabei im Wesentlichen in zwei Schritte:

1. Kantenhervorhebung
2. Erzeugung von Kantenzügen

Zunächst wird das Bild mit einem zweidimensionalen Gaußkern gefaltet:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Dieser sorgt dafür, dass gröbere Störungen und Rauschen beseitigt werden. Das gefilterte Bild wird nun auf Kanten hin untersucht. Während Flächen und Segmente in Bildern meist homogene Grauwerte besitzen, stellen Kanten große Grauwertsprünge dar. Um diese Grauwertsprünge zu detektieren, verwendet man den Sobeloperator, ein Filter, der die partiellen Ableitungen eines Pixels in x- und y-Richtung liefert.
Die Struktur eines 1D Sobelfilters ergibt sich dabei aus der finiten Differenz (hier: Zentraldifferenz) an der Stelle x:

$$\frac{\delta g(x)}{dx} = \frac{g(x + \Delta x) - g(x - \Delta x)}{2 \cdot \Delta x}$$

wobei $\Delta x = 1$. Damit hat der diskrete 1D Filter S die Struktur

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

Erweitert auf einen zweidimensionalen Filter ergibt sich

$$S(x) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, S(y) = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

in x-, bzw. y-Richtung.
Die Anwendung der Filter auf das Ausgangsbild liefert die partiellen Ableitungen G_x und G_y. Aus diesen partiellen Ableitungen lässt sich die Gradientenrichtung d einer Kante berechnen:

$$d(x,y) = arctan(\frac{G_y(x,y)}{G_x(x,y)})$$

Anschließend wird aus den partiellen Ableitungen ein Bild der absoluten Kantenstärke berechnet:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}$$

Im nächsten Schritt wird eine Technik names “Non-maximum suppression” angewandt, um sicherzustellen, dass Kanten nicht breiter als ein Pixel sind. Dabei wird für jedes Pixel dessen 8-Nachbarschaft untersucht. Befindet sich in der Nachbarschaft des zu untersuchenden Pixels ein Pixel, dass einen höheren Grauwert aufweist, so wird der Grauwert des zu untersuchenden Pixels auf o gesetzt, es sei denn, das Pixel mit dem größeren Grauwert befindet sich entlang der Gradientenrichtung.

Im letzten Schritt des Verfahrens werden die Pixel zu Kantenzügen zusammengefasst. Dabei verwendet man zwei Schwellwerte $L_1 < L_2$. Zunächst wird nach einem Pixel gesucht, dessen Grauwert größer als L_2 ist. Dieses Pixel wird zum Startpixel des Kantenzugs erklärt. Nun wird die Kante in beiden Richtungen nach Pixel mit einem Grauwert größer L_1 abgesucht, welche dem Kantenzug hinzugefügt werden. Werden keine Pixel gefunden, die diese Bedingung erfüllen, bricht die Suche ab, und des wird ein neues Startpixel gesucht. Das Verfahren endet, sobald kein unmarkiertes Pixel mit einem Grauwert größer L_2 gefunden wird.

Als Resultat des Verfahrens entsteht ein Bild, das eine Menge von Pixeln enthält, die idealerweise genau die Kantenpixel des Ausgangsbilds enthält.

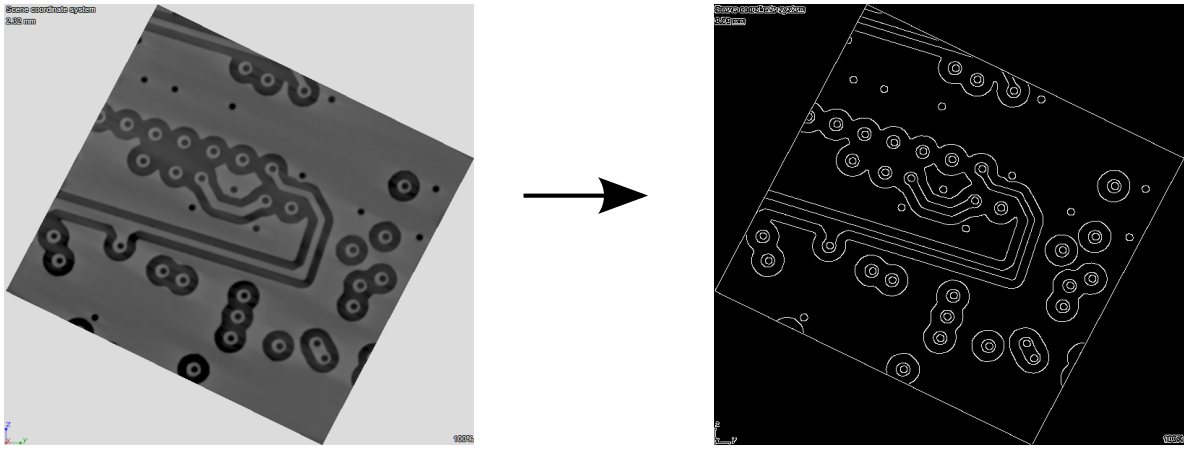

Abbildung 4.1: Resultat nach der Anwendung des Canny-Edge-Detektors

4.0.6 Posterisation

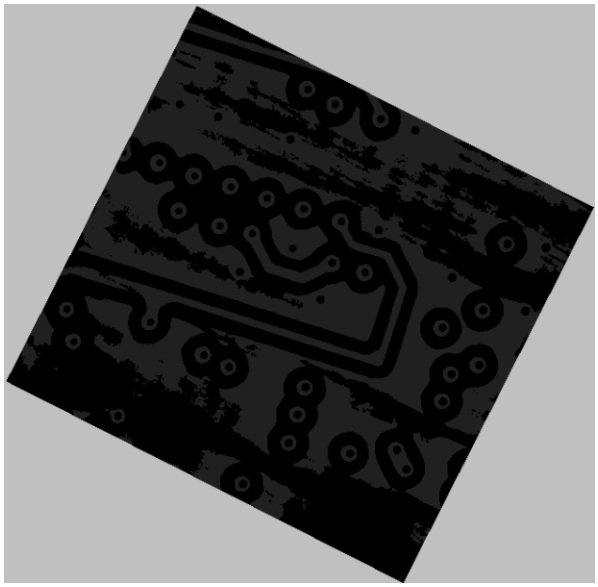

Abbildung 4.2: Posterisation eines 2D-Slices mit Artefakten im Hintergrund

Posterisierte Bilder benutzen eine eingeschränkte Farbpalette. Es gibt keine kontinuierlichen Farbverläufe, alle Farbübergänge sind abrupt. Durch Posterisation können die Bilddaten für die Objekterkennung vorbereitet werden. So kann beispielsweise anschließend durch ein einfaches Template-Matching nach Objekten gesucht werden.
Durch die Eigenschaften der Slices bietet sich die Posterisation als Verfahren für die Separation von Objekten und Hintergrund an. So können beispielsweise auf qualitativ guten Slices

nach der Posterisation durch Template Matching die meisten Bohrpunkte erkannt werden. Das große Problem der Posterisation sind die nicht perfekten CT-Scans. Die Platinen liegen meistens nicht perfekt ausgerichtet in der Aufnahme und somit kommt es vor allem im idealerweise gleichmäßigen Hintergrund zu leichten Artefakten wie z.B. Farbverläufen, die die Posterisation so stark stören, dass das Ergebnis häufig nicht sinnvoll weiterverwerdent werden kann.
Es gibt mehrere Möglichkeiten ein Bild zu posterisieren, eine davon ist die sogenannte Farbreduktion. Dafür werden Algorithmen benutzt, die die n besten Farben für das zu posterisierende Bild ermitteln. Das gebräuchlichste Vorgehen betrachtet die Farbreduktion als ein Cluster-Problem im dreidimensionalen Raum, wobei die Achsen die drei Farbkanäle rot, grün und blau repräsentieren.

Median-Schnitt-Algorithmus

Der Median-Schnitt [Kru94] ist ein Sortierverfahren für n-dimensionale Daten und der meistbenutzte Farbreduktions-Algorithmus. Er unterteilt die Farbwerte anhand des Medians iterativ in Gruppen ähnlicher Werte. Nach der Lokalisation der Gruppen werden deren Farbwerte typischerweise gemittelt um den repräsentierenden Farbwert zu erhalten. Bei der Farbreduktion wird dies so oft wiederholt bis die Anzahl der Gruppen der Anzahl der gewünschten Farben entspricht. Der Median wird mit Hilfe der euklidischen Distanz berechnet:

$$d(a,b) = \sqrt{(a_R - b_R)^2 + (a_G - b_G)^2 + (a_B - b_B)^2}$$

4.0.7 Floodfill-Algorithmus

Floodfill ist ein Algorithmus, der in multi-dimensionalen Arrays zusammenhängende Felder erkennen und färben kann. Ob Felder zusammenhängen ist abhängig davon, ob sie identische Eigenschaften aufweisen. Bei den hier verwendeten Bilddaten wird dafür die Farbe der Pixel/Voxel verwendet. Ausgehend von einem Startpunkt durchsucht der Algorithmus "flutartig" das Array und ändert die Suchrichtung, sobald eine Eigenschaft nicht mehr erfüllt ist.
Der Algorithmus betrachtet ausgehend von einem Startpunkt rekursiv alle benachbarten Felder und färbt diese gegebenenfalls. Ausgehend davon wie viele der Nachbarfelder betrachtet werden spricht man z.B. im zweidimensionalen von "fill4" (oben, unten, links, rechts) oder "fill8" (alle Nachbarn).

Alle eingesetzten Verfahren arbeiten auf dem Resultat eines Canny-Edge-Detektors. Zusätzlich wurde bei den Verfahren, die auf einem gefärbten Bild arbeiten, eine Färbung zusammenhängender schwarzer Flächen nach folgendem Prinzip durchgeführt:

```
for x in xrange(0, width, 3): # stepsize ist 3
       for y in xrange(0, height, 3):
              fill8(img, x, y, (0,0,0), randomColor, resimg)
```

Der implementierte "fill8" -Algorithmus entspricht dem Floodfill-Algorithmus mit einer 8-Pixel Nachbarschaft, welcher im vorigen Abschnitt beschrieben wurde. Dabei wurde der Algorithmus derart modifiziert, dass zuerst in der 8er-Umgebung geprüft wurde ob ein weißer Pixel an den zu untersuchenden Pixel angrenzt und anschließend die 4er-Umgebung auf den Stack gelegt wurde, falls die Prüfung negativ ausgefallen ist. Durch diesen Trick wird verhindert, dass der Algorithmus durch kleinere Lücken (bis 2 Pixel) des Canny-Bildes, welche aufgrund von von fehlerhafter Erkennung entstehen, hindurch läuft.

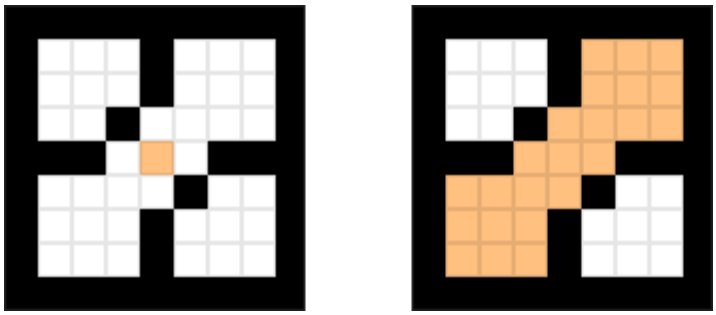

Abbildung 4.3: fill4 färbt ausgehend vom Mittelpunkt die zentrale, weiße Fläche.

4.0.8 Houghtransformation

Die Houghtransformation [BFRR95] baut auf der Kantendetektion auf: Sie sucht Formen, die von Kantenpunkten gebildet werden. Die Houghtransformation ist sehr allgemein verwendbar; Formen können im einfachsten Fall Geraden oder Kreise sein, es können aber auch Ellipsen und andere geometrische Figuren detektiert werden, es handelt sich bei der Houghtransformation also um eine **modellbasiertes** Verfahren.
Mit der verallgemeinerten Houghtransformation können sogar beliebige Formen gefunden werden. Die Houghtransformation benutzt ein einfaches Grundprinzip: Man untersucht alle Kantenpunkte auf Hinweise auf eine gegebene Form, die man detektieren möchte. Diese Hinweise werden in einem Akkumulatorraum (Parameterraum) gespeichert. Nachdem alle Kantenpunkte untersucht wurden, wertet man die gesammelten Hinweise im Akkumulatorraum aus.
Die Untersuchung der Kantenpunkte und die Dimension des Akkumulatorraums hängen dabei von der Form ab, die man detektieren will.

Houghtransformation zur Erkennung von Geraden

Wendet man auf ein Bild die üblichen Kantenfilter an (Sobel-Operator, Canny-Edge-Detektor) zeigt sich, dass diese Verfahren zwar eine Menge von Punkten liefern, die auf Kanten, bzw. Geraden liegen, die Gruppierung dieser Punkte zu echten Kanten allerdings fehlt. Hier setzt die Houghtransformation an, indem es ein zusammenhängendes Geradenstück auf einen Punkt im Akkumulatorraum abbildet.

Eine Gerade, beschrieben durch die Gleichung $y = mx + b$ würde also auf einen Punkt im zweidimensionalen Akkumulatorraum, der durch m und b aufgespannt wird, abgebildet werden. Dadurch können allerdings senkrechte Geraden, die keine Steigung besitzen, nicht mehr korrekt in den Akkumulatorraum abgebildet werden.
Daher verwendet man die Hessesche Normalform zur Darstellung der Geraden:

$$r = x \cdot cos(\theta) + y \cdot sin(\theta),\ \theta \in [0,\ 2\pi]$$

Die Gerade ist also die Menge aller Punkte (x, y), die diese Gleichung erfüllen. θ sei der Winkel zwischen der y-Achse und der Geraden, $r \in D$ der Abstand der Geraden zum Ursprung, wobei D die Diagonallänge des Bildes sei.

Der zugehörige Akkumulatorraum A wird damit von θ und r aufgespannt.

Zunächst wird das Ausgangsbild mit einem Gradientenoperator gefiltert (z.B. mit dem Sobel-Filter), woraus sich eine Matrix der Gradientenstärke $G(x, y)$ und der Gradientenrichtung $\phi(x, y)$ berechnen lässt (siehe Canny-Edge-Detektor).
Auf $G(x, y)$ wird nun ein Schwellwert G^* angewandt, um die N Pixel zu erhalten, die nicht durch kleinere Grauwertschwankungen im Ausgangsbild verursacht wurden. Man erhält die Menge

$$M = \{(x, y) | G(x, y) \geq G^*\},\ |M| = N.$$

Im nächsten Schritt "votiert" jedes Element $(x, y) \in M$ für einen Punkt (θ, r) im Akkumulatorraum, indem der Wert um 1 erhöht wird:

$\forall (x, y) \in M$:

1. Bilde $r = x \cdot cos(\theta) + y \cdot sin(\theta)$ aus x, y und $\phi(x, y)$
2. Erhöhe (θ, r) um 1

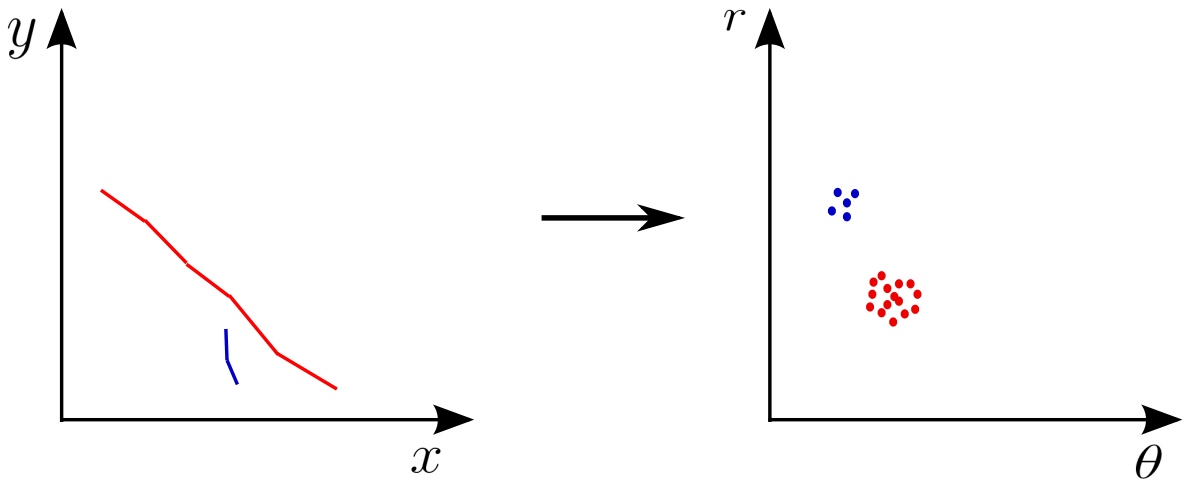

Abbildung 4.4: Abbildung der Geraden in den Akkumulatorraum

Alle Punkte, die auf einer Geraden liegen, votieren somit für den gleichen Punkt in A. Allerdings werden Punkte einer Geraden aufgrund von Ungenauigkeiten nicht exakt für den selben Punkt votieren, sondern einen Cluster bilden. Daher genügt es nicht, im Akkumulatorraum nach großen Werten zu suchen, sondern es ist zusätzlich eine Clusteranalyse notwendig, um alle Punkte auf der Geraden zu „erwischen“.

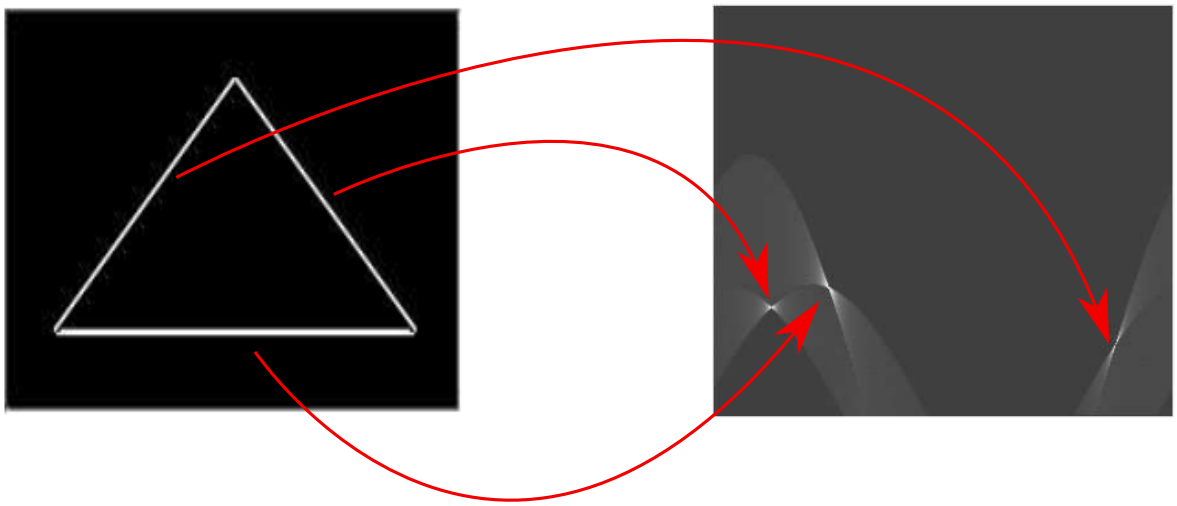

Abbildung 4.5: Dreiecksgeraden werden in den Akkumulatorraum projeziert

Houghtransformation Erkennung von Kreisen

Die Houghtransformation [Haro9] kann auch zur Erkennung von Kreisen eingesetzt werden. Auch hier werden „Hinweise“, auf die gesuchte Form im Akkumulatorraum gesammelt. Ausgehend von einem Kantenbild wird jeder Pixel einer Kante als von Kreisen mit beliebigem (oder festgelegtem) Radius erzeugt angesehen. Die Transformation in den Akkumulatorraum funktioniert so, dass man dort alle Kreismittelpunkte einträgt, die Kreise erzeugen könnten, auf denen der Pixel liegen würde. Es wird also der entsprechende Punktpixel im Akkumulatorraum um 1 erhöht.
Falls nebenbei zu jedem Kantenpixel die zugehörigen Kantenrichtung bekannt ist, kann diese Information genutzt werden und es bleiben nur noch zwei möglich Kreise übrig.
Wenn nun die Punkte im Kantenbild einen Kreis repräsentieren, ist an der zum Mittelpunkt gehörenden Stelle im Akkumulatorraum ein besonders hoher Wert eingetragen, da dort sehr viele Kantenpixel des Kreises für den Mittelpunkt abgestimmt haben. Die Maxima im Akkumulatorraum repräsentieren also die Kreismittelpunkte.

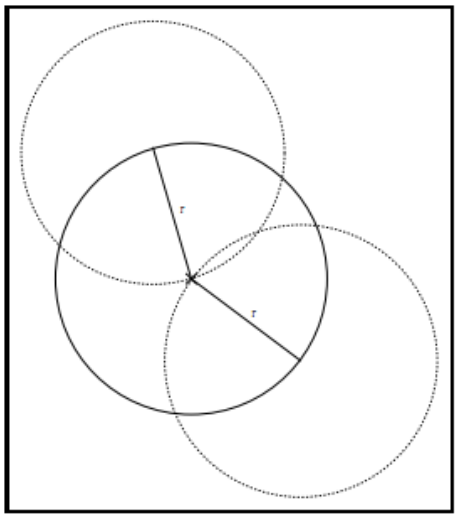

Abbildung 4.6: Suche nach Kreise mit bekanntem Radius:
Ein Kreis mit Radius r_0 kann durch seinen Mittelpunkt (x_0, y_0) charakterisiert werden. In der Abbildung sieht man einen Kantenpunkt (dargestellt durch ein Kreuz). Die Mittelpunkte der Kreise mit Radius r_0, zu denen dieser Punkt gehören könnte (zwei davon sind gestrichelt eingezeichnet), liegen auf einem Kreis mit Radius r_0 um ihn herum.

Die ersten zwei Dimensionen des Hough-Raums entsprechen hier also denen des Bildraums, da die (x,y)-Koordinaten in die Lage des Kreismittelpunktes beschreiben. Zusätzlich dazu ist laut der Kreisgleichung $x^2 + y^2 = r^2$ der Radius r der dritte Parameter, der beachtet werden sollte, wenn man nach Kreisen mit beliebigen Radius sucht. Falls letzteres nicht der Fall ist, kann der Parameter weggelassen werden, was im zweidimensionalen Fall einen einfach darzustellenden und anschaulichen Hough-Raum liefert.
Anschließend muss natürlich auch hier, wie auch im Fall von Geraden, der Akkumulatorraum auf lokale Maxima untersucht werden.

5 Selektion gut detektierbarer Bilder

Um aus dem beschriebenen Bilderstack möglichst gut detektierbare Einzelbilder zu erhalten, werden hier verschiedene Verfahren vorgestellt, die weitestgehend automatisch eine Auswahl von Bildern erstellt, die für die weitere Objekterkennung gut geeignet sind.
Da der gescannte Mikrochip aus vier Leiterbahnschichten besteht, muss der beim Scannen entstehende Bilderstack in vier Intervalle eingeteilt werden, welches jeweils eine Leiterbahnschicht repräsentiert.

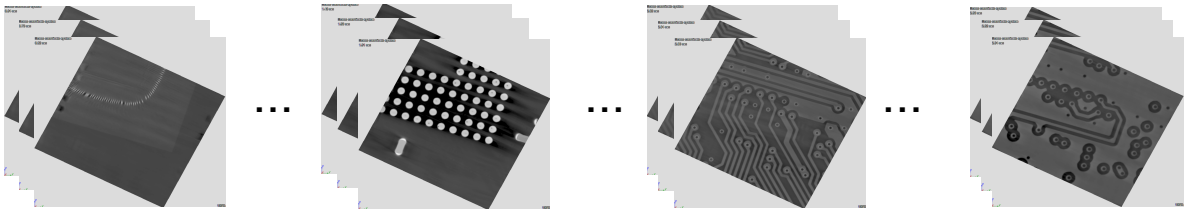

Abbildung 5.1: Einteilung des Bilderstacks in vier Intervalle

Ziel der unten vorgestellten Verfahren besteht darin, um für jedes Intervall das Bild zu finden, dass die Leiterbahnschicht am genausten repräsentiert. Die aus den Verfahren gewonnen Bilder werden anschließend weiterverwendet, d.h. es werden die 2D Objekterkennungsverfahren darauf angewandt und bewertet, wie gut die ausgewählten Bilder für die einzelnen Verfahren geeignet sind, und ob für jedes Intervall tatsächlich das am besten detektierbare Bild gefunden wurde.

Um die aufgeführten Verfahren miteinander vergleichen zu können, wird für jedes Intervall das „Best Match“ vor dem Vergleich ausgewählt. Dabei handelt es sich um jenes Bild, in dem die Objekte am besten sichtbar sind. Für die vier Intervalle sind dies folgende Bilder:

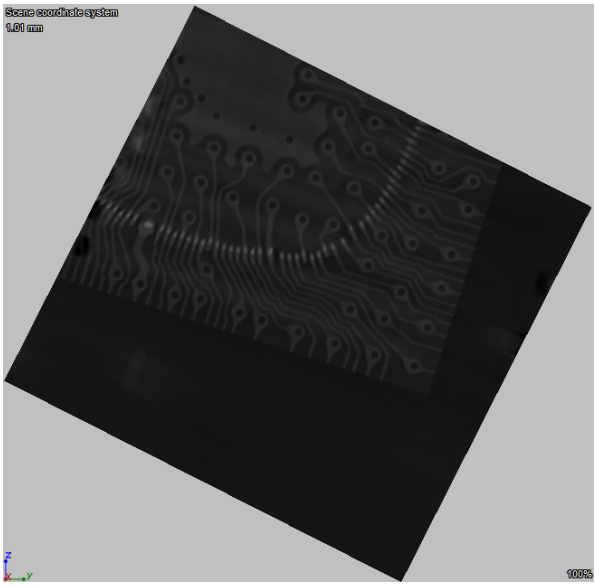

Abbildung 5.2: Best Match des ersten Intervalls

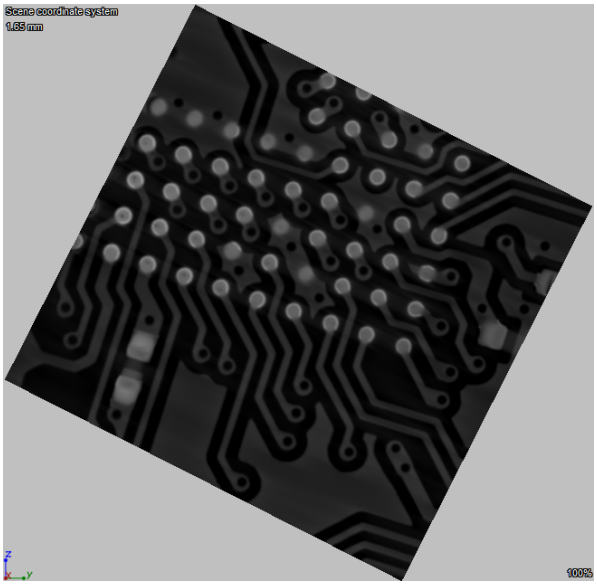

Abbildung 5.3: Best Match des zweiten Intervalls

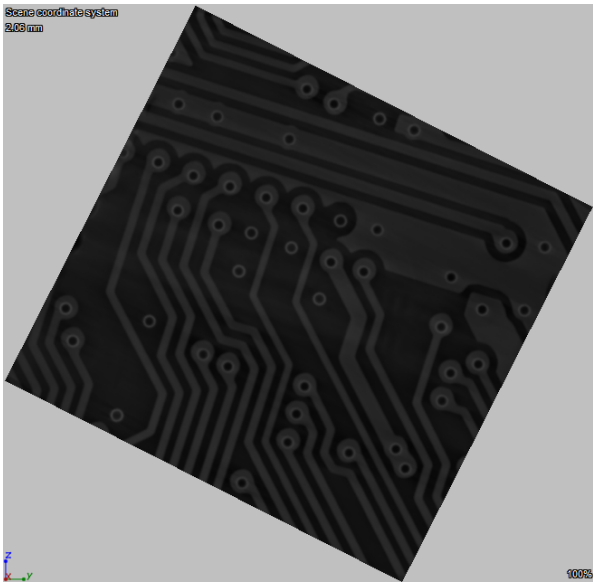

Abbildung 5.4: Best Match des dritten Intervalls

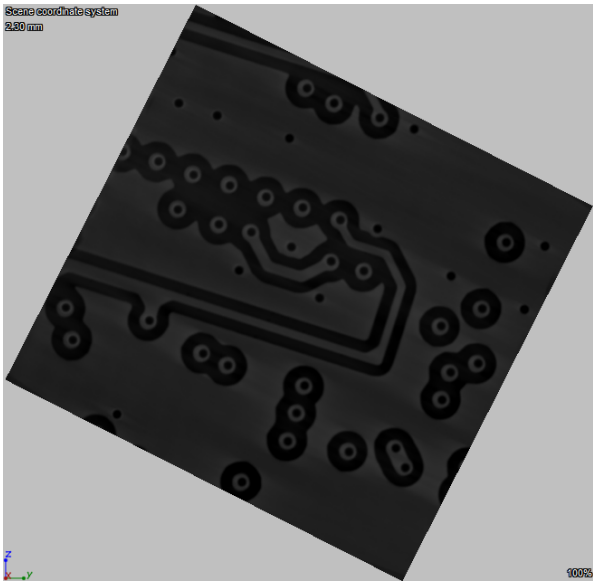

Abbildung 5.5: Best Match des vierten Intervalls

Diese Best Matches haben im Bilderstack jeweils die Indizes

1. Intervall 1: $Index27$
2. Intervall 2: $Index61$

3. Intervall 3: *Index*83
4. Intervall 4: *Index*96

Die Ergebnisse der einzelnen Verfahren werden mit diesen Best Matches verglichen, indem die durschnittliche Distanz der Ergebnisse zu den Best Matches im Bilderstack ermittelt wird.

5.1 Binärbilder

Bei diesem Verfahren werden zunächst alle Bilder in Binärbilder umgewandelt. Als binärer Schwellwert wird dabei der Wert 70 gewählt, d.h. alle Pixel, deren Grauwert größer als 70 ist, wird der Grauwert 255 zugewiesen, allen Pixeln die einen Grauwert kleiner, bzw. gleich 70 besitzen, wird der Grauwert 0 zugewiesen. Anschließend wird das Bild ausgewählt, welches die maximale Anzahl an weißen Pixeln (Pixel mit dem Grauwert 255) besitzen.

Die Idee, die hinter diesem Verfahren steckt, ist, dass Bilder, die eine möglichst große Anzahl von Pixeln mit maximalen Grauwert besitzen, viele Objekte enthalten, die detektiert werden können.

5.1.1 Ergebnis

Die Ergebnisse des Verfahrens für die vier Intervalle:

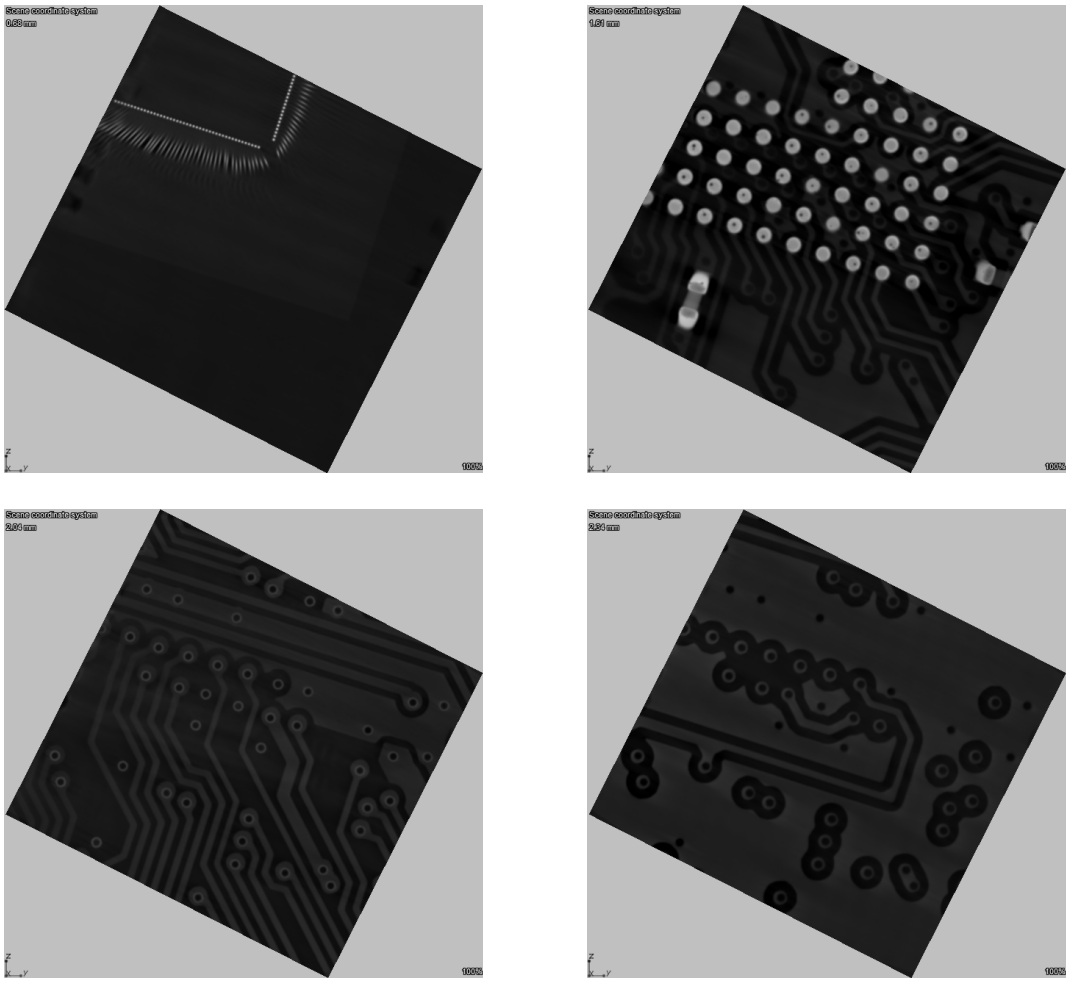

Abbildung 5.6: Ergebnisse des Binärbildverfahrens

Für die einzelnen Intervalle sind dies die Bilder mit folgenden Indizes:

1. Interval l1: *Index* 9 , Differenz zum Best Match: 18
2. Intervall 2: *Index* 59, Differenz zum Best Match: 2
3. Intervall 3: *Index* 82, Differenz zum Best Match: 1
4. Intervall 4: *Index* 98, Differenz zum Best Match: 2

Dies führt zu einer durchschnittlichen Distanz von **5,75** vom Best Match.

5.2 Canny-Edge-Bilder

Bei diesem Verfahren werden die einzelnen Bilder der Intervalle zunächst mit dem Canny-Edge Operator in die entsprechenden Kantenbilder überführt. Anschließend werden die Kantenpixeln in den Kantenbildern gezählt, wobei für jedes Intervall das Bild ausgewählt wird, dass die meisten Kantenpixel besitzt.

Der Ansatz hierbei ist, dass Bilder, die viele einzelne Objekte enthalten, mehr Kantenpixel liefern, als Bilder, in denen wenige Objekte enthalten sind.

5.2.1 Ergebnis

Die Ergebnisse des Verfahrens für die vier Intervalle:

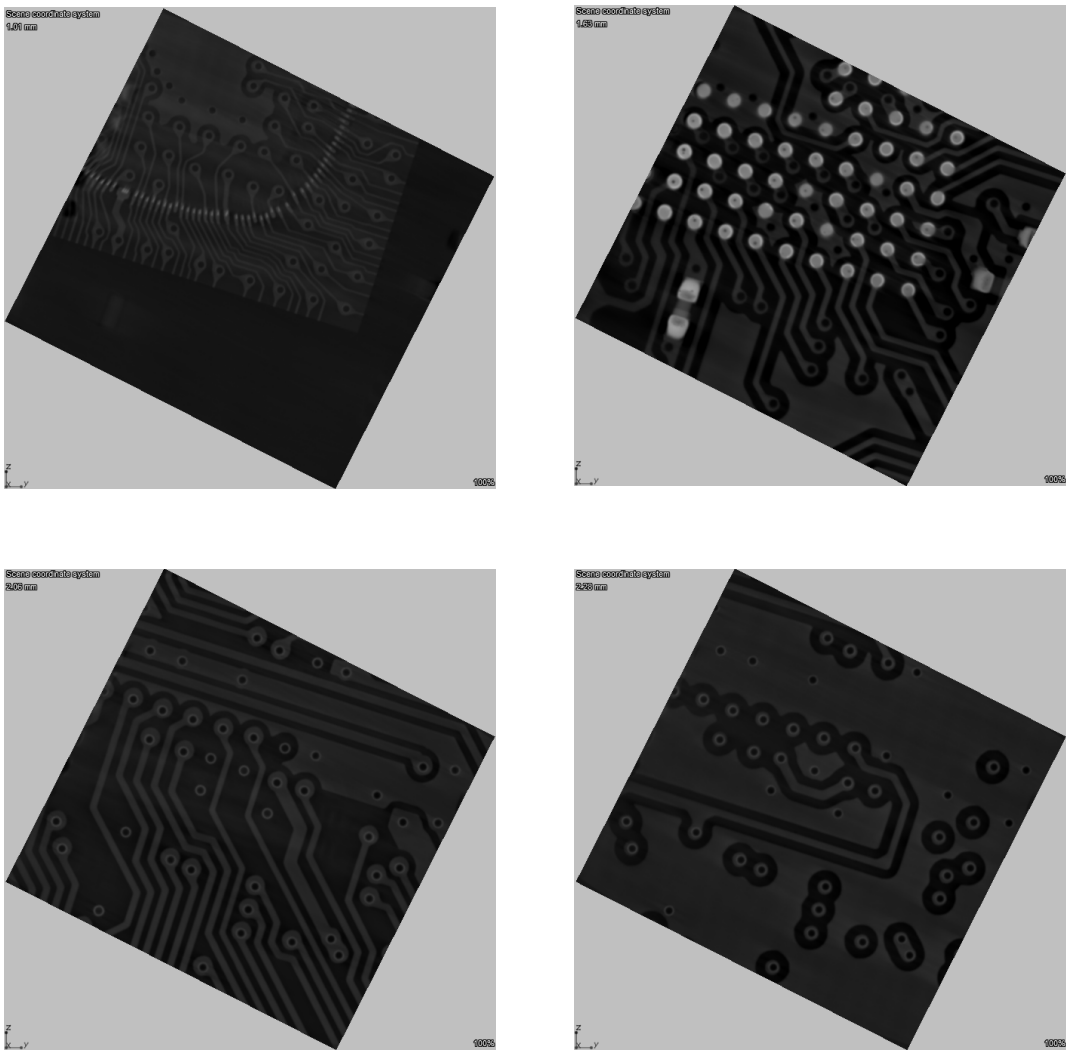

Abbildung 5.7: Ergebnisse des Canny-Edge-Verfahrens

Die zugehörigen Katenbilder:

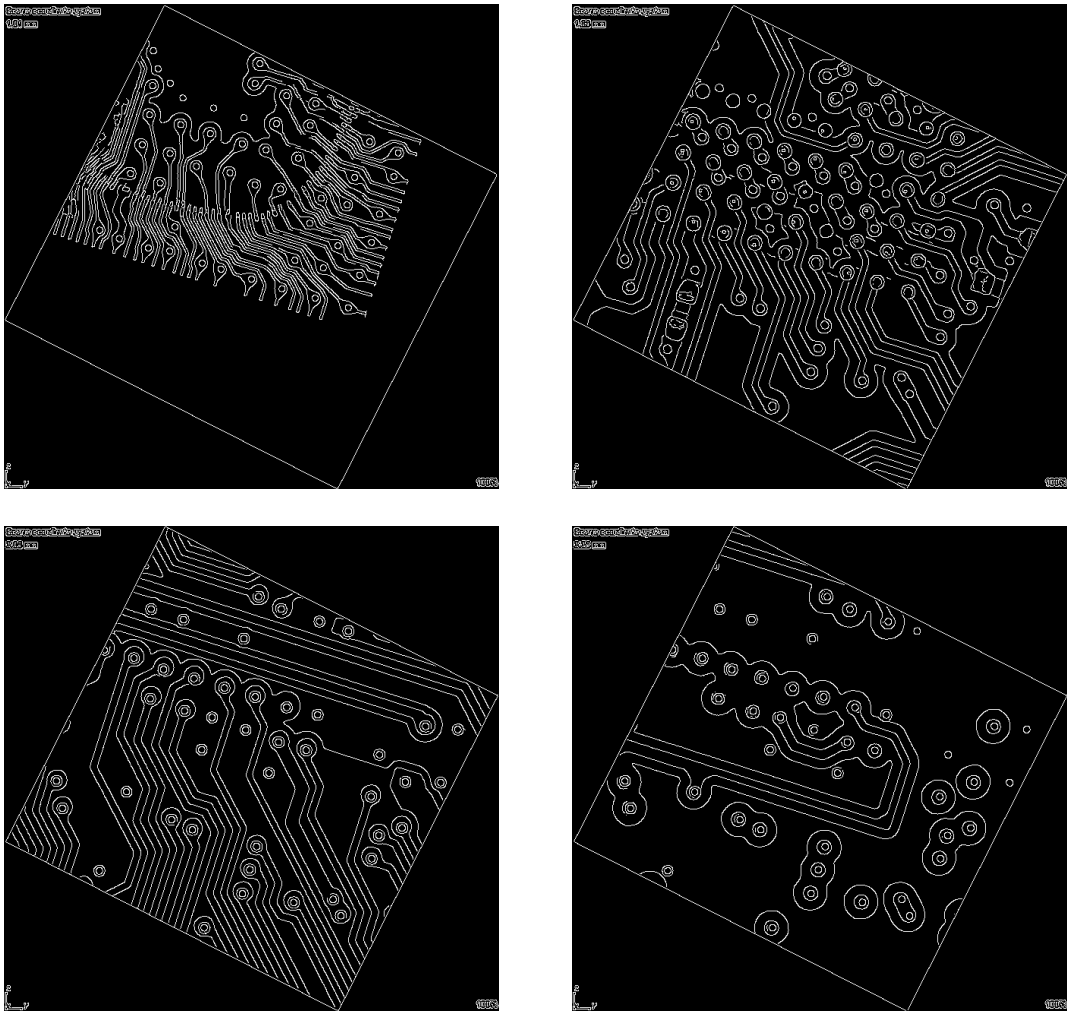

Abbildung 5.8: Kantenbilder der Ergebnisse des Canny-Edge-Verfahrens

Für die einzelnen Intervalle sind dies die Bilder mit folgenden Indizes:

1. Intervall 1: *Index* 27, Differenz zum Best Match: 0
2. Intervall 2: *Index* 60, Differenz zum Best Match: 1
3. Intervall 3: *Index* 83, Differenz zum Best Match: 0
4. Intervall 4: *Index* 95, Differenz zum Best Match: 1

Dies führt zu einer durchschnittlichen Distanz von **0,5** vom Best Match.

5.3 Hough-Transformation

Dieses Verfahren ziehlt darauf ab, die Bilder in den Intervallen zu finden, die möglichst gut detektierbare Kantenverläufe, und somit möglichst gut erkennbare Leiterbahnen enthalten. Dafür werden aus den Bildern mit Hilfe der Houghtransformation die Kanten extrahiert und anschließend gezählt. Das Maximum für jedes Intervall ist jenes Bild, welches die meisten zählbaren Kanten enthält.

5.3.1 Ergebnis

Die Ergebnisse des Verfahrens für die vier Intervalle:

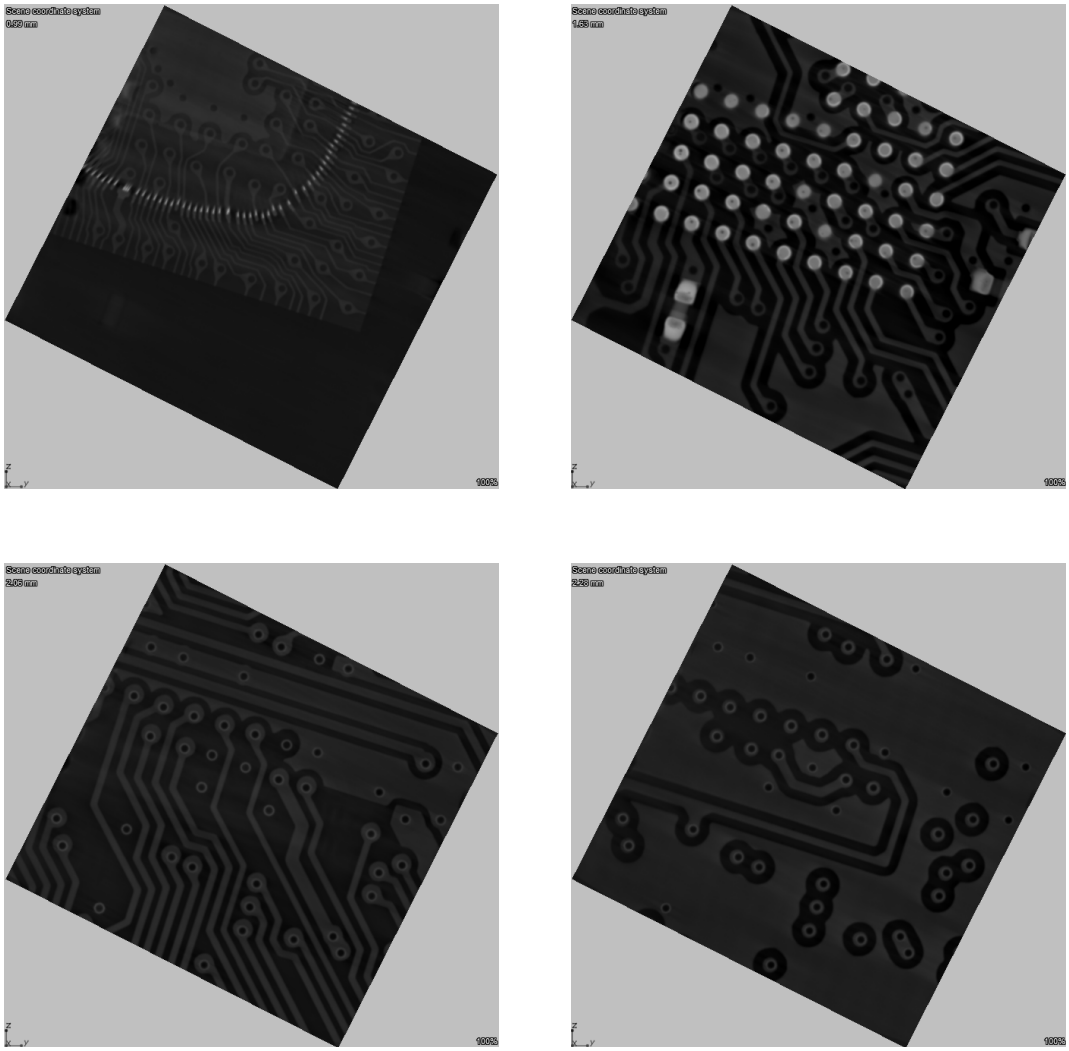

Abbildung 5.9: Ergebnisse des Houghtransformationsverfahrens

Für die einzelnen Intervalle sind dies die Bilder mit folgenden Indizes:

1. Intervall 1: *Index* 26, Differenz zum Best Match: 1
2. Intervall 2: *Index* 60, Differenz zum Best Match: 1
3. Intervall 3: *Index* 83, Differenz zum Best Match: 0
4. Intervall 4: *Index* 95, Differenz zum Best Match: 1

Dies führt zu einer durchschnittlichen Distanz von **0,75** vom Best Match.

5.4 Vergleich

Es ist deutlich zu erkennen, dass das Binärbildverfahren deutlich schlechtere Ergebnisse (durchschnittliche Distanz: 5,75) liefert, als das Canny-Edge-Verfahren (durchschnittliche Distanz: 0,5) und das Houghtransformationsverfahren (durchschnittliche Distanz: 0,75). Dies liegt vor allem daran, dass Störpixel, bzw. für die weitere Verarbeitung irrelevante Informationen, in Form von nicht defninierten Objekten einen starken Einfluss auf die Menge der weißen Pixel nehmen. Beim Canny-Edge-Verfahren werden primär „echte“ Objekte, also Objekte, die aus zusammenhängenden Kanten bestehen, extrahiert, während Störpixel bei der Überführung der Bilder in Kantenbilder gefiltert werden.
Ein minimal schlechteres Ergebniss liefert das Houghtransformationsverfahren, da dieses Verfahren eine gute Detektierbarkeit der Leiterbahnen berücksichtigt, Bohrpunkte allerdings außer Acht lassen.

6 2D - Verfahren

6.1 SIFT

SIFT (Scale-invariant feature transform [Low04]) ist ein von David Lowe 1999 vorgestelltes Verfahren zur Extraktion von lokalen Merkmalen aus Bildern. Die mit diesem Verfahren gefundene Merkmale haben die Eingeschaft, dass sie robust gegenüber Rotation, Translation und Skalierung sind, und damit zuverlässig in anderen Bilder wiedererkannt werden können. Um Objekte in Bildern mit SIFT erkennen und lokalisieren zu können, sind also zwei Schritte nötig:

1. Extraktion und Beschreibung von Merkmalen (Features) des gesuchten Objekts
2. Lokalisation der Merkmale im Suchbild

6.1.1 Extraktion und Beschreibung von Merkmalen (Features) des gesuchten Objekts

Der Algorithmus zur Extraktion und Beschreibung der Merkmale besteht dabei aus vier Verarbeitungsstufen:

Ermittlung potentieller Merkmale in DoG-Pyramiden

Um Merkmale zu ermitteln, die robust gegenüber Skalierung sind, kommt das Verfahren der DoG (Difference of Gaussians) Pyramiden zum Einsatz. Dabei werden aus dem Ausgangsbild zunächst n Gaußpyramiden berechnet. Eine Pyramide besteht dabei aus fortlaufend stärker geglätteten Bildern des Ausgangsbildes g. Zur Glättung kommt dabei ein Gaußfilter G zum Einsatz:

$$g(x,y) * G_\vartheta(x,y) = g(x,y) * (\frac{1}{\sqrt{2\pi\vartheta^2}} \cdot e^{-\frac{x^2+y^2}{2\vartheta^2}})$$

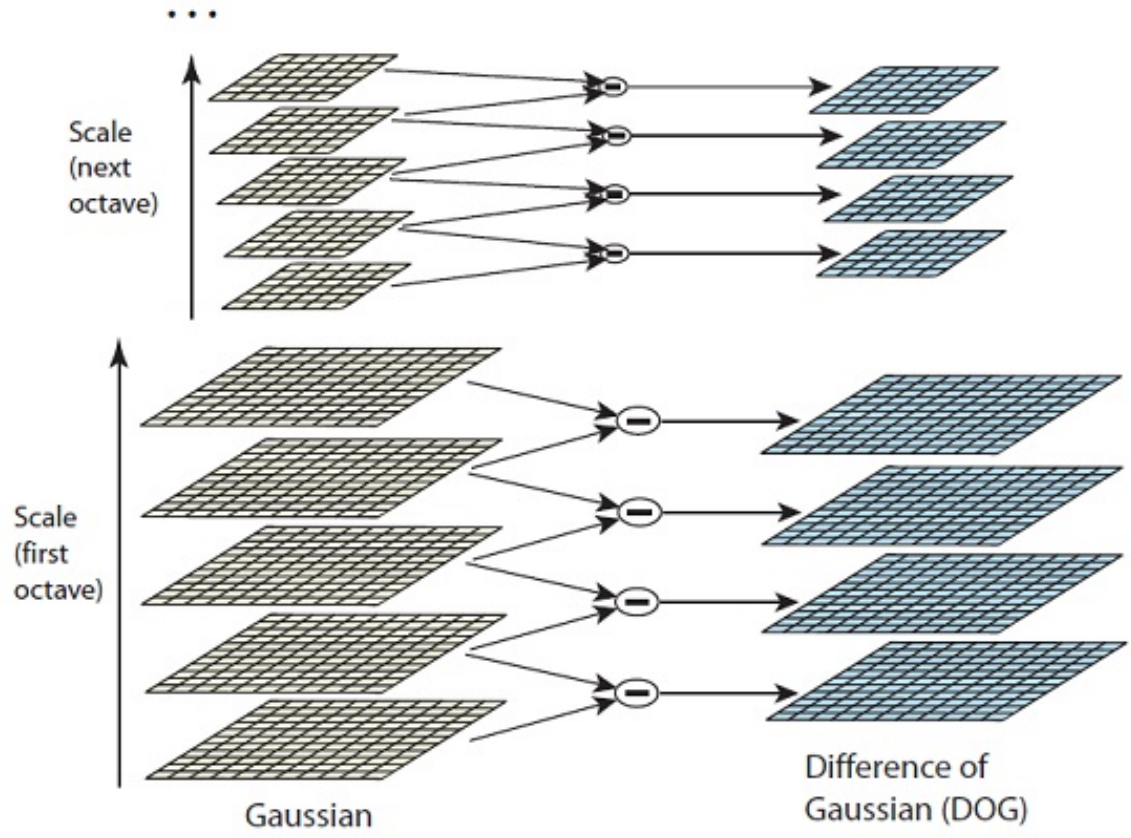

Abbildung 6.1: DoG-Pyramiden

Im Anschluss wird das letzte Bild der Pyramide um 50% verkleinert, und daraus durch erneute fortlaufende Glättung mit dem Gaußfilter eine neue Pyramide erzeugt. Je zwei benachbarte Bilder einer Gaußpyramide werden nun voneinander subtrahiert. Aus den Resultaten entstehen dabei die DoG-Pyramiden:

Die dadurch erzeugten DoG-Pyramiden werden nun auf minimale und maximale Pixelwerte untersucht. Ein Maximum ist gefunden, wenn der Grauwert eines Pixels größer als der seiner 26 Nachbarn ist. Nachbarschaft eines Pixels ergibt sich dann aus seinen acht Nachbarn der selben Ebene, sowie aus den jeweils neun Nachbarn der benachbarten Ebenen in der DoG-Pyramide. Die Suche nach Minima erfolgt auf die selbe Art und Weise. Die Information, auf welcher Skalierung die potentiellen Merkmalspunkte liegen, wird dabei ebenfalls gespeichert.

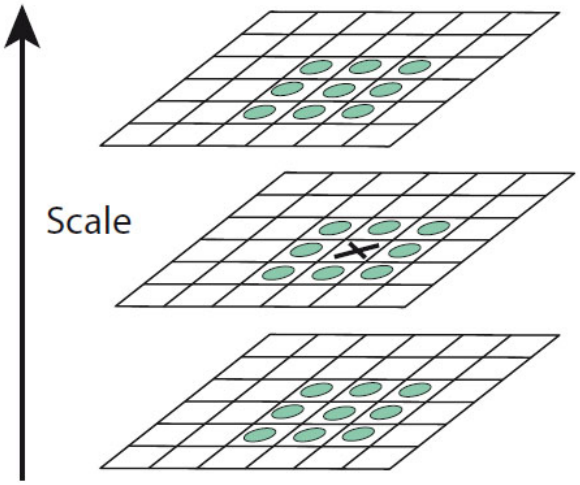

Abbildung 6.2: Nachbarschaft eines Pixels

Filterung und Lokalisation potentieller Merkmalspunkte

Das oben genannte Verfahren liefert neben den robusten Merkmalspunkten eine große Menge von instabilen, für die weitere Verarbeitung nicht zu gebrauchende Merkmale. Daher werden die gefundenen Merkmalspunkte anhand von Stabilitätskriterien gefiltert. Im ersten Schritt werden dabei alle Merkmalspunkte entfernt, die einen DoG-Wert von weniger als 0.03, und somit einen relativ niedrigen Kontrast besitzen. Merkmalspunkte, die auf Ecken liegen sind „prägnanter“ (und somit staibler) als solche, die auf einer Kante liegen, daher werden alle Merkmalspunkte entfernt, die auf einer Kante, aber nicht auf einer Ecke liegen. Dies geschieht unter Anwendung der Hesse-Matrix.

Bestimmung der Hauptorientierungen

Um Invarianz der verbleibenden Merkmalspunkte gegenüber Rotation zu erreichen, wird für jeden Merkmalspunkt dessen Hauptorientierung berechnet. Dafür nutzt man das gaußgefilterte Bild, welches der Skalierung des zu untersuchenden Merkmalspunktes am nächsten kommt. In diesem Bild werden nun innerhalb einer festen Region um den Merkmalspunkt herum die Gradientenlängen $m(x, y)$ und die Gradientenorientierungen $\theta(x, y)$ bezüglich eines Punktes $g(x, y)$ berechnet, wobei

$$m(x,y) = \sqrt{(g(x+1,y) - g(x-1,y))^2 + (g(x,y+1) - g(x,y-1))^2}$$

und

$$\theta(x,y) = tan^{-1} \cdot \frac{g(x+1,y) - g(x-1,y)}{g(x,y+1) - g(x,y-1)}$$

Die so ermittelten Gradientenorientierungen werden nun anhand ihrer Gradientenlängen gewichtet. Dadurch haben Gradientenrichtungen mit großer Gradientenlänge einen größeren Einfluss auf die Hauptorientierung als Gradientenrichtungen mit niedriger Gradientenlänge. Danach werden die Gradientenorientierungen zusätzlich anhand ihrer Entfernung zum Merkmalspunkt gewichtet, um Gradientenrichtungen, die sich näher am Merkmalspunkt befinden stärker zu gewichten.

Aus den gewichteten Gradientenorientierungen wird nun ein Orientierungshistogramm erstellt. Dieses Histogramm ist in 36 Winkelbereiche eingeteilt und hat somit eine Klassenbreite von $10\,^{\circ}$. Jede Gradientenorientierung wird dabei anhand ihrer Gewichtung an der passenden Stelle im Histogramm aufaddiert.
Nach der Erstellung des Histogramms kann aus diesem die Gradientenlänge m_{max} abgelesen werden (Winkelbereich mit der größten Summe). Die Hauptorientierung des Merkmalspunktes setzt sich dabei aus m_{max}, sowie der zugehörigen Gradientenorientierung θ_{max} zusammen. Für den Fall, dass eine weitere Orientierung mit der Gradientenlänge $m_i > 0,8m_{max}$ existiert, wie es bei Eckpunkten häufig der Fall ist, wird an der Stelle (x,y) ein weiterer Merkmalspunkt mit der Hauptorientierung (m_i, θ_i) erstellt.

6.1.2 Lokalisation der Merkmale im Suchbild

Wurden nun im ersten Schritt die robusten Merkmale des gesuchten Objekts extrahiert, können diese im Suchbild wiedererkannt werden. Dies geschieht, in dem man die extrahierten Merkmale des Objekts mit denen im Suchbild auf Übereinstimmung hin untersucht.
Der dafür am häufigsten verwendete Ansatz ist der Vergleich anhand des euklidischen Abstands der Merkmalsvektoren.

$$e = \sqrt{\sum_{i=1}^{n}(V_{1i} - V_{2i})}$$

6.2 Einsatz von SURF zur Erkennung von Bohrpunkten

Um in den Beispieldatensätzen Bohrpunkte auf möglichst effiziente Art und Weise erkennen zu können, wird hier SURF (*SpeededUpRobustFeatures*), eine leicht veränderte Variante des SIFT-Verfahrens, verwendet.
Der Unterschied zum hier vorgestellen SIFT Verfahrens besteht darin, dass statt der Gaußfilter Mittelwertfilter zum Einsatz kommen. Dadurch wird das Verfahren signifikant beschleunigt, ohne die Erkennungsrate nennenswert zu beeinflussen. [BTG05]

Um Bohrpunkte mithilfe des SURF Verfahrens zu detektieren, wird zunächst ein Modell des Bohrpunktes auf dessen Merkmale hin untersucht. Dazu wird ein Template eines Bohrpunktes aus einem Bild im Bilderstack ausgeschnitten.

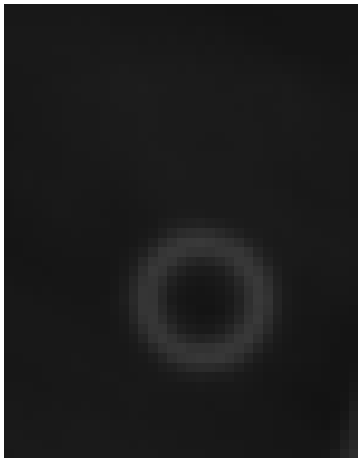

Abbildung 6.3: Modell eines Bohrpunktes (Vergrößert)

Auf dieses Template wird nun der besprochene SURF Algorithmus angewandt, um die Merkmalsvektoren des Bohrpunktes zu extrahieren. Das Template wurde dabei bewusst so klein gewählt, dass das Verfahren genau einen Merkmalsvektor liefert, welcher den Bohrpunkt repräsentiert:

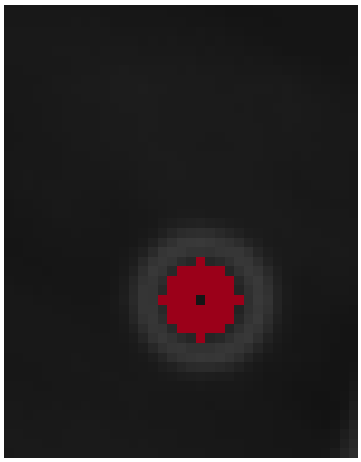

Abbildung 6.4: Bohrpunkt mit eingezeichnetem Merkmalsvektor (Vergrößert)

Im nächsten Schritt werden die Merkmalsvektoren im Suchbild extrahiert, wobei wieder der SURF-Algorithmus zum Einsatz kommt, und eine Liste mit Merkmalsvektoren liefert. Zur Illustration werden auch hier die Orte der Merkmalsvektoren in das Suchbild gezeichnet:

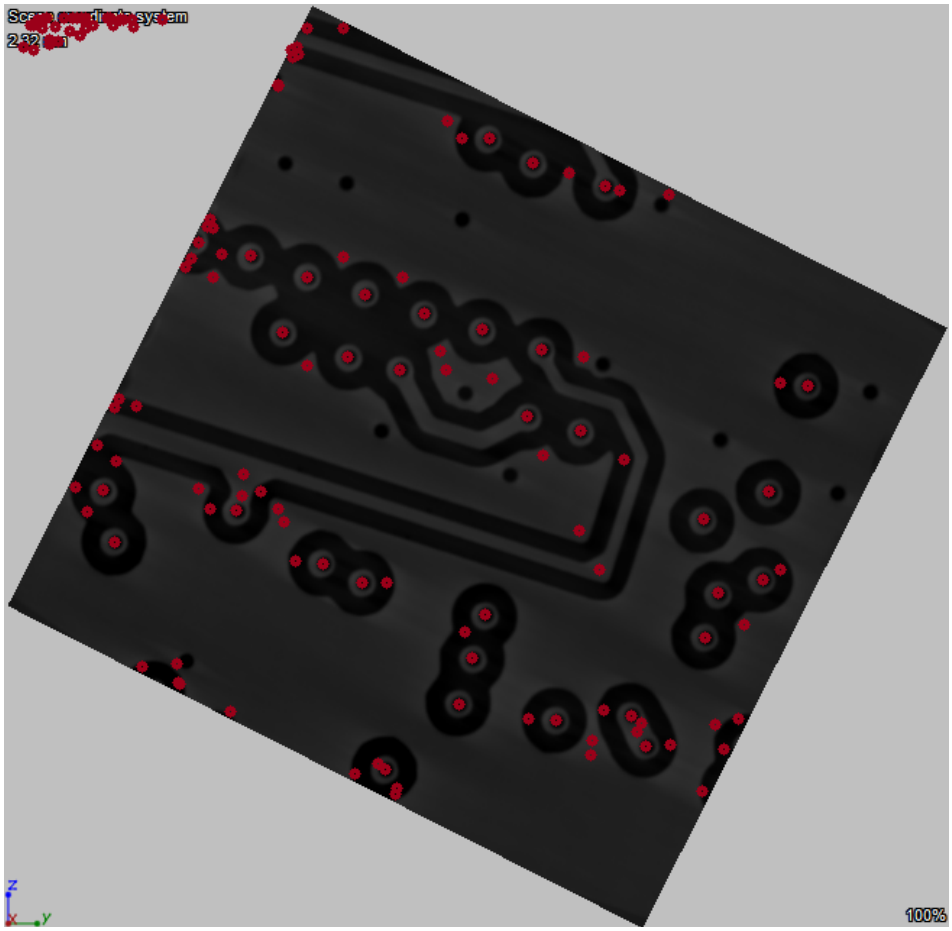

Abbildung 6.5: Suchbild mit eingezeichneten Merkmalsvektoren

Anschließend wird für den Merkmalsvektor des Templates nach Übereinstimmungen in der Liste der Merkmalsvektoren des Suchbildes gesucht, d.h. es wird nach Merkmalsvektoren gesucht, die dem des Templates möglichst ähnlich sind. Ähnlich bedeutet hier, dass die Komponenten zweier Merkmalsvektoren eine hohe Übereinstimmung haben, was genau dann der Fall ist, wenn beide Merkmalsvektoren einen geringen Abstand im Merkmalsraum haben.
Daher wird der euklidische Abstand des Merkmalsvektors des Templates mit jedem Merkmalsvektor der Liste berechnet. Ist der Abstand dabei kleiner als $0,45$ im Merkmalsraum, wird eine Übereinstimmung der beiden Merkmalsvektoren angenommen.
Dabei werden folgende übereinstimmende Merkmalsvektoren gefunden:

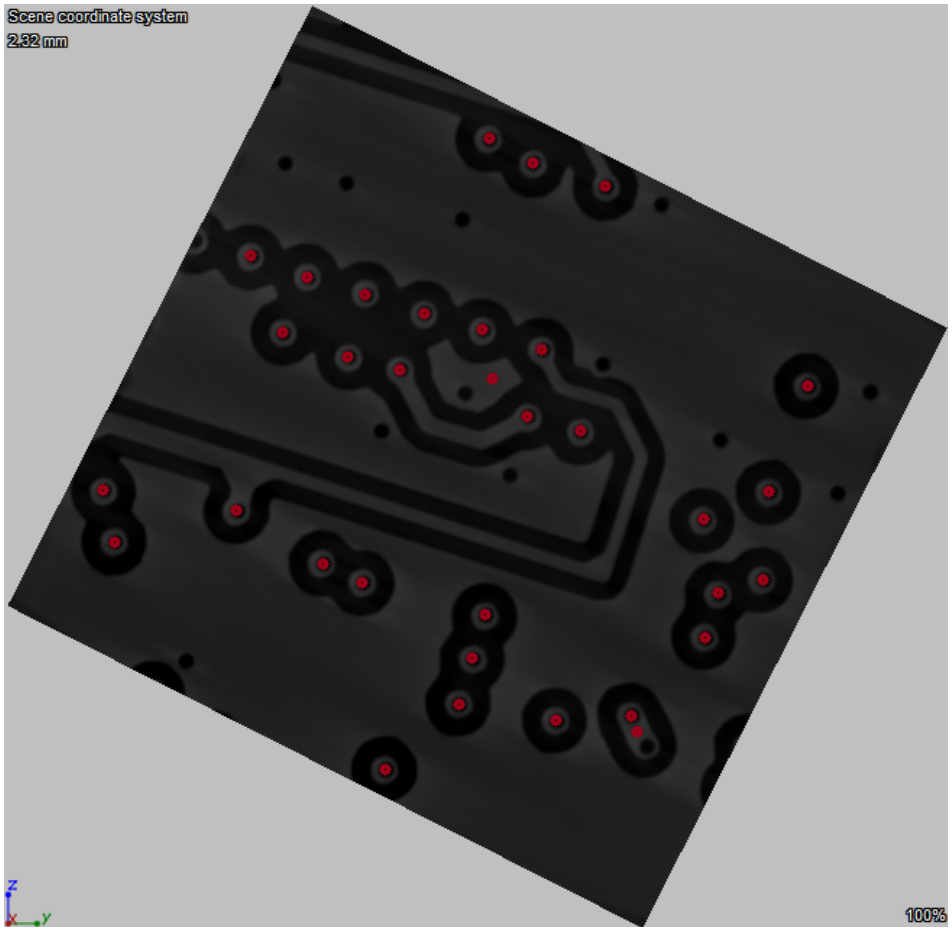

Abbildung 6.6: Suchbild mit eingezeichneten übereinstimmenden Merkmalsvektoren

Wie deutlich zu sehen ist, werden Bohrpunkte, die visuell dem Template entsprechen (Bohrpunkt mit äußerer Isolierschicht) fehlerfrei erkannt. Allerdings werden die Bohrpunkte ohne Isolierschicht nicht erkannt. Dieser Umstand lässt sich durch eine Modifikation des Suchverfahrens verbessern.

6.3 Erweiterung der Merkmalssuche

Wie im vorigen Abschnitt gezeigt wurde, erkennt das Verfahren des Matchings mit von SIFT-Merkmalen bei einem einzelnen Bild nicht zufriedenstellend, da die Merkmale der Bohrpunkte ohne Isolierschicht nicht eine zu große Distanz zum Merkmalsvektor des Templates im Merkmalsraum haben.
Allerdings zeigt sich, dass in Bilder, die sich im Bilderstack nahe am im vorigen Kapitel untersuchten Bild befinden, diese Bohrpunkte erkannt werden. Daher beschränkt man hier die

Merkmalssuche nicht auf ein einzelnes Bild, sondern erweitert die Suche auf eine Teilmenge des Bilderstacks. Dabei werden jeweils die 20 Bilder des Bilderstacks untersucht, die dem Ausgangsbild am nächsten sind.
Die in diesen 20 Bildern gefundenen Merkmalsvektoren werden anschließend gemerged, d.h. Merkmalsvektoren die mehrfach vorkommen, werden verworfen, so dass von ihnen jeweils nur ein Merkmalsvektor übrig bleibt. Dieses Merging wird wieder mithilfe des euklidischen Abstands im Merkmalsraum vollzogen: Ist der Abstand zweier Merkmalsvektoren im Merkmalsraum geringer als eine bestimmt Schwelle, werden diese als gleich angesehen und ein Merkmalsvektor wird verworfen. Die Orte der verbleibenden Merkmalsvektoren werden wieder im Bild markiert:

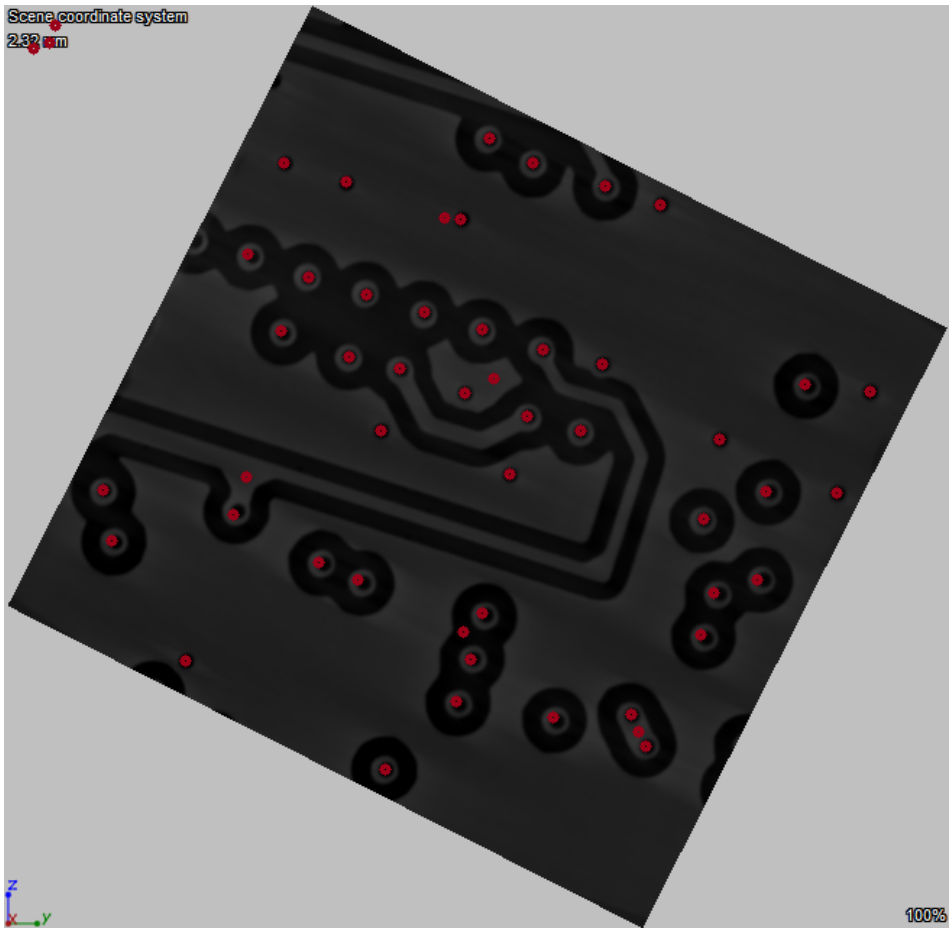

Abbildung 6.7: Suchbild mit eingezeichneten gemergeten Merkmalsvektoren

6.3.1 Bewertung:

Bis auf einiger Fehler in Form von Merkmalsvektoren, die keine Bohrpunkte beschreiben, wurde im getesteten Bild auf diese Art und Weise alle Bohrpunkte, egal ob mit oder ohne

Isolierschicht, erkannt. Das Verfahren wurde jedoch nur für das gezeigte Testbild eingehend untersucht, daher können keine Aussagen über die Fehlerhäufigkeit bei Bildern mit einer komplexeren Struktur getroffen werden.
Es hat sich gezeigt, dass die Kombination der einzelnen Szenenbilder bemerkenswerte Verbesserungen erbracht hat. Dieses Verfahren eignet sich somit vor allem dann, wenn mehrere unterschiedliche Bilder der gleichen Szene existieren (was in dem zu untersuchenden Bilderstack der Fall ist).

6.4 Template Matching

Template Matching ist ein Verfahren, bei dem ein prototypisches Modell einer Struktur im Bild gesucht wird. Das Template ist dabei selbst ein kleines Bild, welches wie ein Filterkern über das Bild wandert.

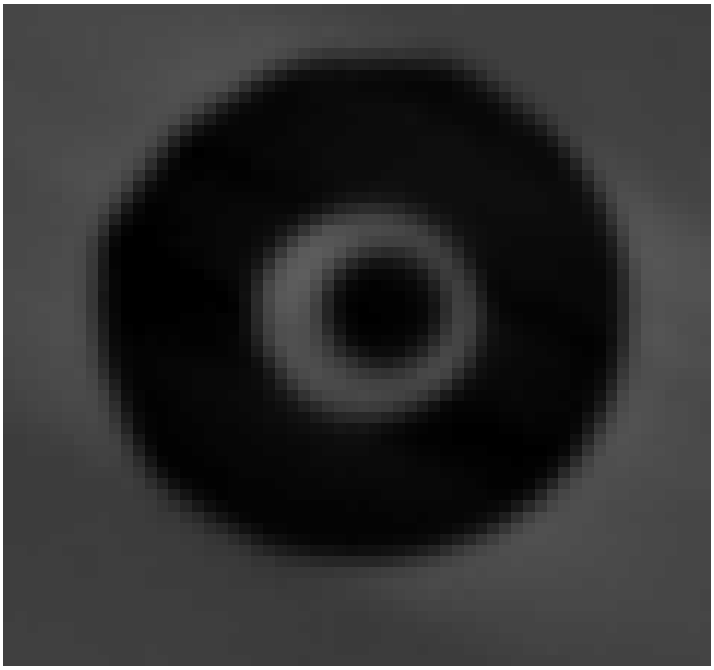

Abbildung 6.8: Template eines Bohrpunktes (Vergrößert)

Dabei wird in jedem Punkt (x, y) ein Änhlichkeitsmaß des Templates gegenüber dem Bild berechnet. Ein häufig verwendetes Änglichkeitsmaß ist dabei **Mean Absolute Difference (MAD)**. Dieses Änhlichkeitsmaß bezeichnet die mittlere Differenz der Grauwerte des Bildes g und des Templates T:

$$MAD(x,y) = \frac{1}{M \cdot N} \sum_{ij} |g(x+i, y+j) - T(i,j)|$$

Befindet sich bei der Suche das Template genau über der gesuchten Struktur, ist $MAD(x, y)$ minimal, während bei keiner Übereinstimmung des Templates und des Bildausschnittes $MAD(x, y)$ groß ist. Dadurch sind im resultierenden Bild, in dem das Änhlichkeitsmaß abgebildet wird, lokale Minima die Orte, in denen sich die Struktur des Templates befindet.

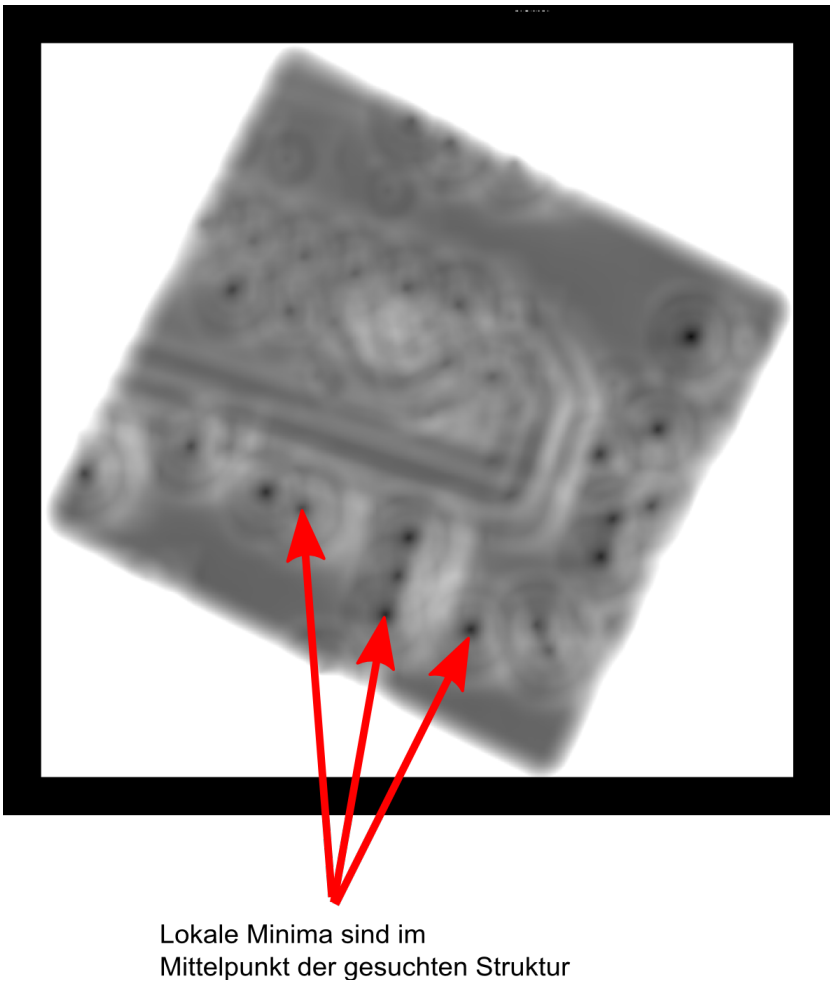

Abbildung 6.9: Lokale Minima

Um die genaue Position der Bohrpunkte zu ermitteln, müssen diese Minima detektiert werden.
Der einfachste Ansatz dafür ist ein Schwellwert zu benutzen, so dass nach der Schwellwertbildung lediglich die lokalen Minima übrig bleiben.

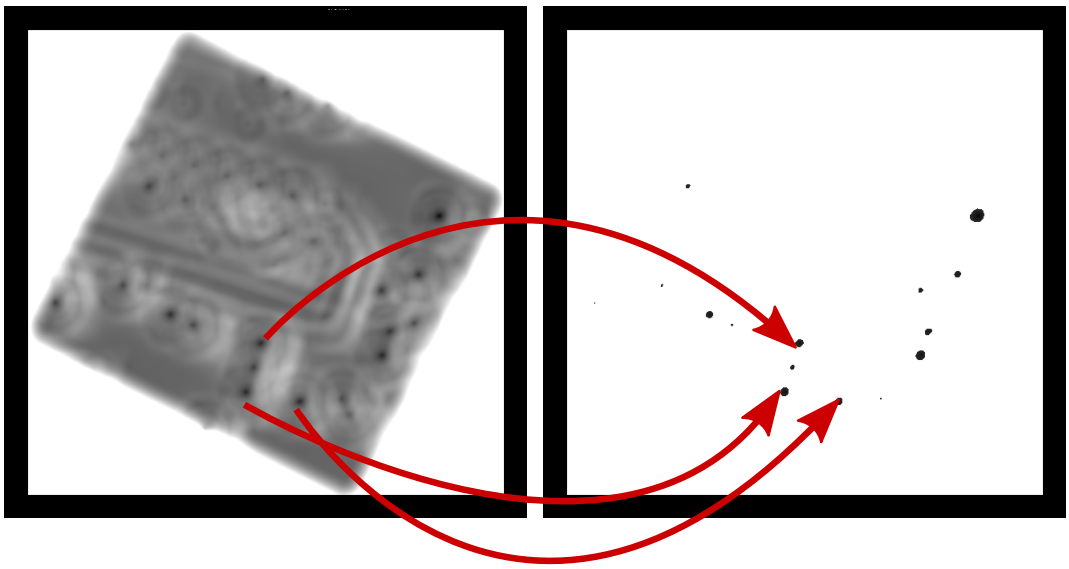

Abbildung 6.10: Lokale Minima nach der Schwellwertbildung

Ein Problem bei diesem Verfahren ist den richtigen Schwellwert zu treffen. Ein zu niedriger Schwellwert lässt die lokalen Minima verschwinden. Ein zu hoher Schwellwert führt zu einem „auslaufen“ der lokalen Minima in die angrenzenden Regionen.

Abbildung 6.11: Auslaufen der lokalen Minima bei zu hohem Schwellwert

6.4.1 Bewertung:

Die Unterschiede bei den lokalen Minima, welche durch die Wahl des Templates, Helligkeits- und Farbunterschiede im Suchbild oder geringer Formunterschiede der Bohrungen und ihrer Umgebung entstehen, machen eine automatisierte Suche nach den lokalen Minima sehr schwierig. Sowohl das Template als auch der Algorithmus zur Extraktion der lokalen Minima muss heuristisch optimiert werden, um bessere Ergebnisse zu erzielen. Da es im Umfang der Fachstudie nicht gelungen dies effektiv zu atomatisieren, muss der Templatematching-Algorithmus für die hier betrachteten Daten als ungeeignet eingestuft werden.

6.5 Einsatz der Houghtransformation zur Erkennung von Bohrpunkten

Da gleichartige Bohrungen wohl auf der kompletten Platine den gleichen Radius (mit einer kleinen Toleranz) besitzen, kann auf 2d-Slices explizit nach Kreisen mit diesem Radius gesucht werden. Dies ergibt für Houghtransformation für Kreise mit Radius = 5 beispielsweise folgendes Bild im Akkumulatorraum:

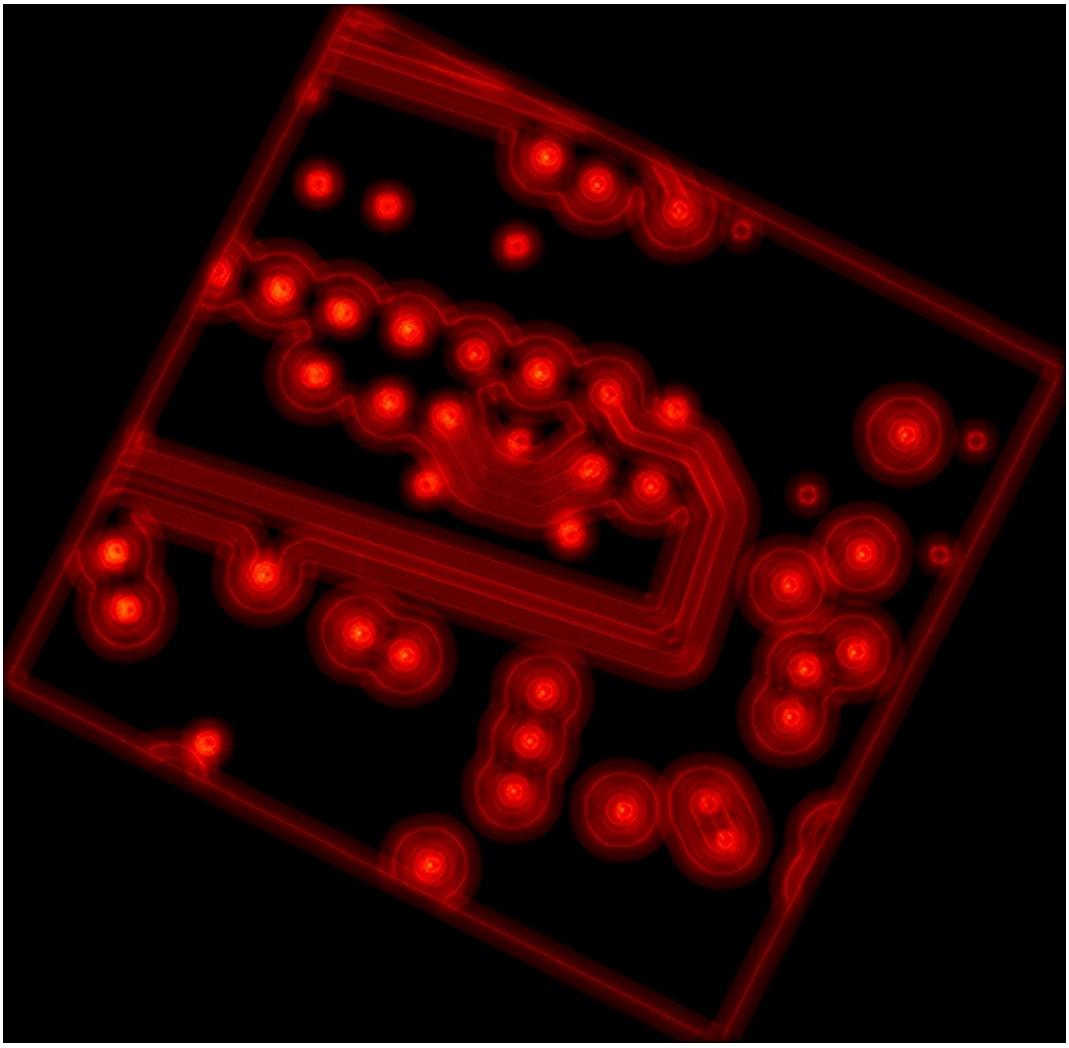

Abbildung 6.12: Akkumulatorraum für Kreise mit Radius = 5

Da die Radien jedoch leicht schwanken, was insbesondere durch die geringe Auflösung bedingt ist, kann beispielsweise noch der Raum für Kreise mit Radius 4 hinzuaddiert werden, um ein klareres Bild zu erhalten.

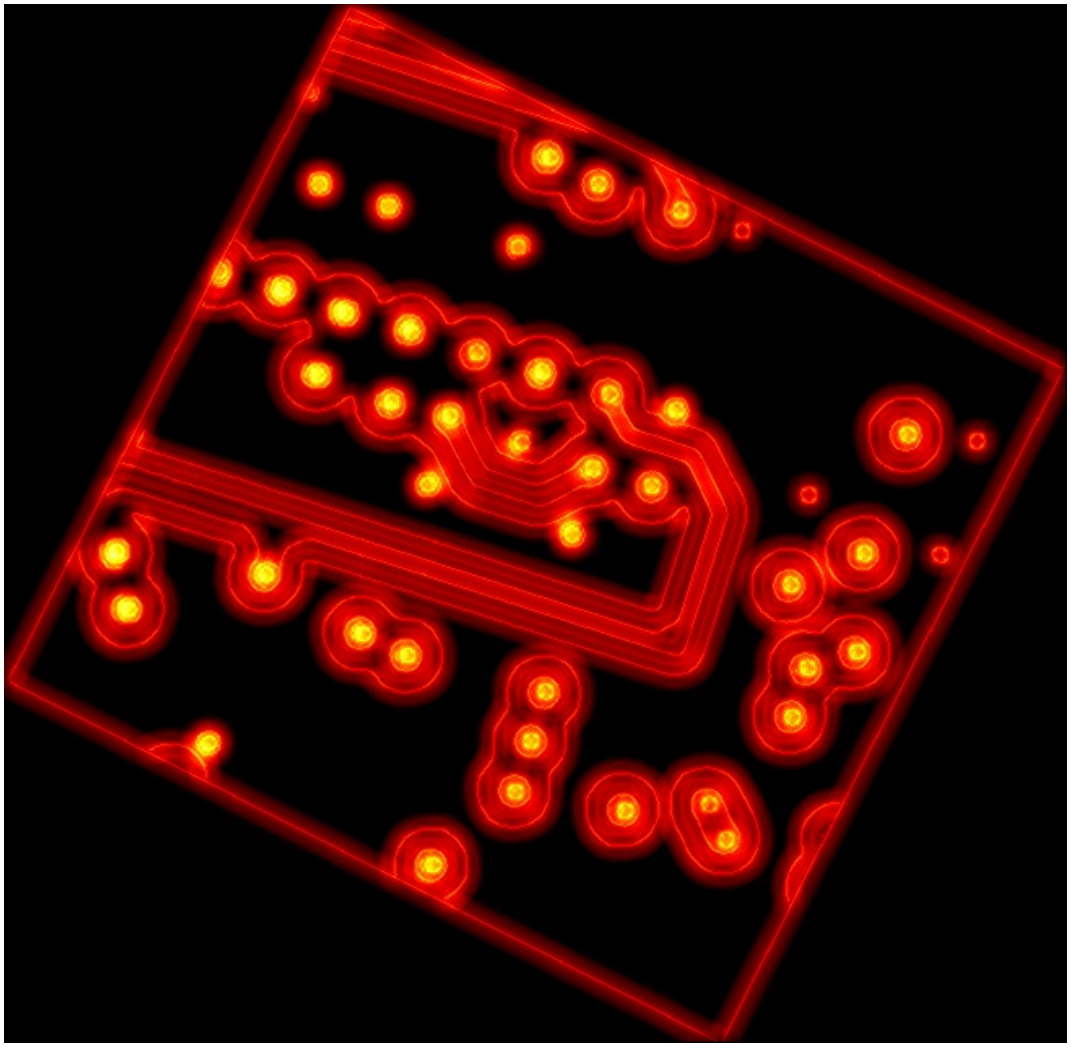

Abbildung 6.13: Akkumulatorraum für Kreise mit r=4 und r=5

In diesem Raum müssen nun die lokalen Maxima gefunden werden. Dies ist keine triviale Aufgabe, da ein lokales Maxima in einem anderen Bildbereich eher zum oberen Durchschnitt gehört. Es bietet sich hier der Einfachheit halber dennoch an alle Pixel oberhalb eines Schwellwertes (z.B $0.6 \cdot globMax$) zu Clustern zusammenfassen und aus diesen jeweils den Mittelpunkt als Bohrung zu speichern.

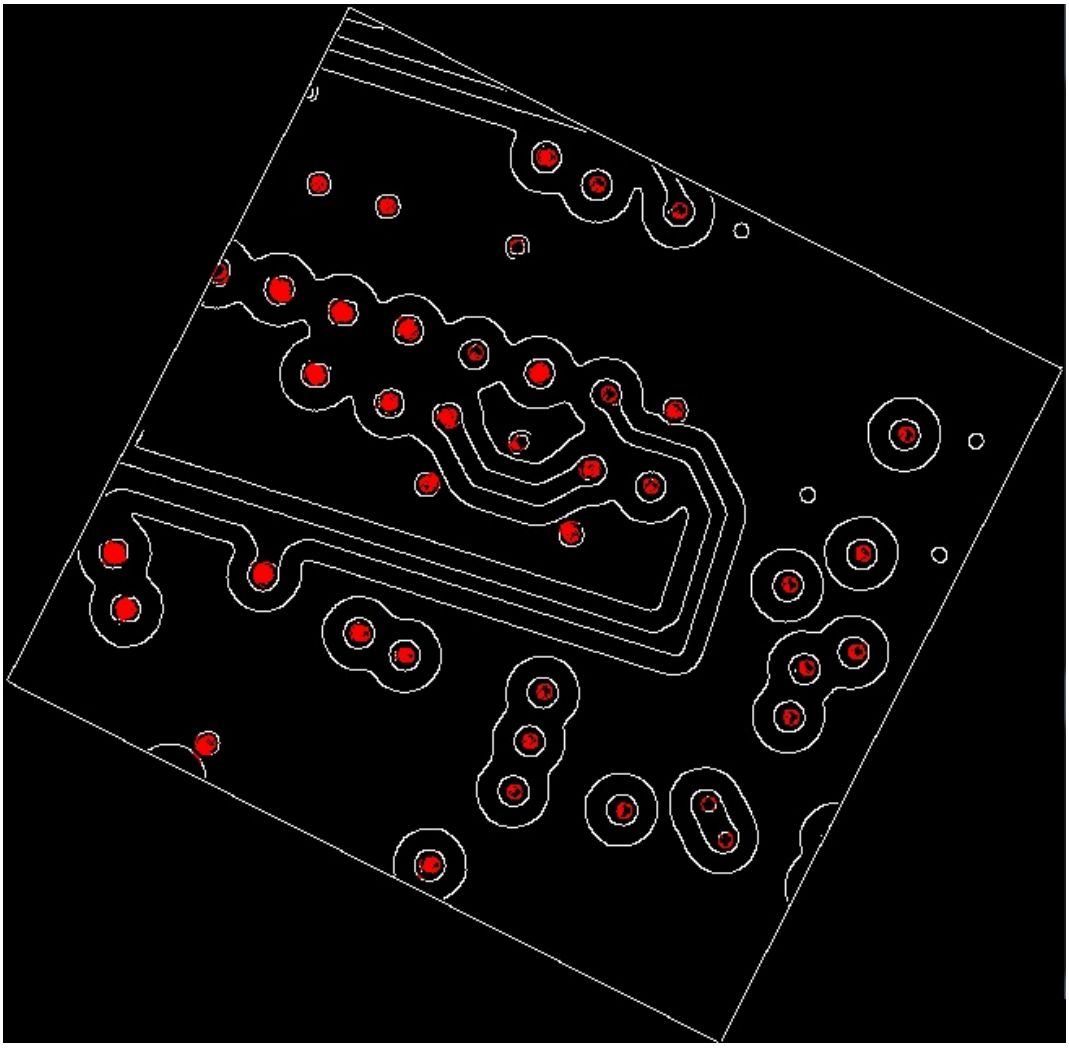

Abbildung 6.14: Pixel mit höherem Wert als $0.6 \cdot globMax$

Wendet man ein solches Schwellwertverfahren an, so müsste der Schwellwert heuristisch optimiert werden, um möglichst viele Punkte zu „erwischen“, da Bohrungen ohne umliegende Kanten (bspw. Isoliermaterial) generell schwächer in den Akkumulatorraum abgebildet werden.
Um dieses Bild zu verbessern, hat man natürlich die Möglichkeit den Akkumulatorraum mit verschiedenen Filtern zu falten, um damit zum Beispiel alle kreisförmigen Maxima zu verstärken.
Ansonsten könnte man beispielsweise verschiedene „Hill-Climbing“-Algorithmen einsetzen, wobei hier zu beachten ist, dass ein lokales Maximum nur dann als Hinweis für eine Bohrung interpretiert werden darf, wenn die Menge zusammenhängender, umgebender Pixel mit ähnlich hohem Wert lokal relativ beschränkt ist. Dadurch werden nur „punktförmige“ Maxima gefunden.

Im Anschluss müssen natürlich noch die einzelnen Cluster jeweils zu Bohrpunkten zusammengefasst werden.

6.5.1 Bewertung:

Die Unterschiede bei den lokalen Minima, welche durch Größen- und Formunterschiede der Bohrungen und ihrer Umgebung entstehen, machen eine automatisierte Suche nach den lokalen Minima sehr schwierig. Da sich die Houghtransformation jedoch nur auf die Resultate des Canny-Edge-Detektores stützt, fallen die Unterschiede der lokalen Minima kleiner aus als beim Template-Matching. Daher ist es auch mit dem einfachen Schwellwert-Verfahren bereits möglich die meisten Bohrungen in der Platine ausfindig zu machen.

6.6 Alternativer Algorithmus zur Erkennung von Bohrpunkten auf Grundlage einer Färbung

Die Houghtransformation für Kreise ist relativ rechenintensiv und schließlich muss der Akkumulatorraum noch ausgewertet werden. Auch das Templatematching arbeitet mit Filterkernen, die so groß sind wie die gesuchten Kreise. Dies motivierte dazu einen schnelleren Algorithmus zu entwickeln, zur Not auch auf Kosten der Universalität. Eine sehr einfache und effiziente Möglichkeit Kreise in einem Canny-Edge Bild zu finden ist die folgende:

procedure FindCircles(Bild img, Radius r, Toleranz t)
  for all Pixel $p \in img$ **do**
    if p ist weiß **then**
      Folge der zugehörigen weißen Linie l ungefähr n Schritte lang
      , wobei $n <= int(2 \cdot \pi \cdot (r + t))$
      for all Mittelpunkte mp **do** // Geschätzt aus der Position von p.
        if $\exists lp \in l : dist(mp, lp) < r - t \vee dist(mp, lp) > r + t$ **then**
          Abbrechen
        end if
        if Die Linie umschließt den Mittelpunkt (z.B. Quadrantencheck) **then**
          l sei ein Kreis; continue in äußerer Schleife
        end if
      end for
    end if
  end for
end procedure

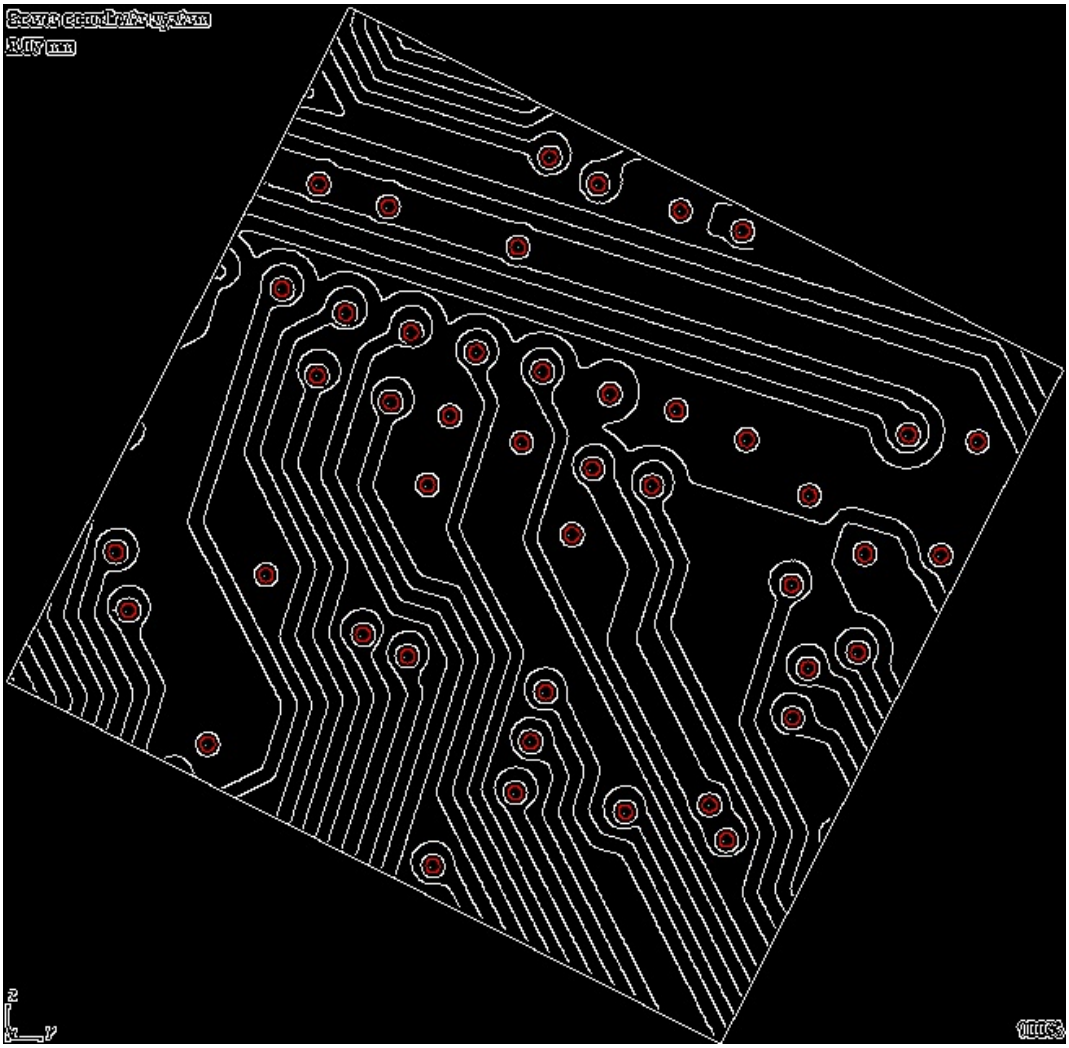

Abbildung 6.15: Ein Resultat des Algorithmus (gefundene Bohrungen sind rot markiert)

6.6.1 Bewertung:

Der Algorithmus ist extrem schnell und erkannte in sämtlichen Testbildern ohne Probleme alle gesuchten Kreise. Falls die Kantendetektion einen Fehler gemacht hat und der Kreis eine kleinere Lücke beinhaltet, so wird der Kreis im allgemeinen dennoch erkannt (Falls die Lücke nicht zu groß ist).
Problematisch könnte selbstverständlich die Verallgemeinerung auf die gleichzeitige Suche von Kreisen beliebiger Radien werden. Diesbezüglich wurden noch keine Anstrengungen unternommen, da hierfür noch keine Notwendigkeit bestand.

6.7 Einsatz der Houghtransformation zur Erkennung von Leiterbahnen

Die Houghtransformation ist nicht direkt geeignet für das Erkennen von Leiterbahnen, da ausschließlich Geradenstücke erkannt werden. Aus diesen lässt sich jedoch nicht schließen, ob es sich um eine Leiterbahn handelt oder nicht (es könnte z.B. auch Teil vom Isoliermaterial sein).
Die Kurven der Bahnen werden nicht erkannt, somit müsste man die Geradenstücke manuell verlinken und anschließend noch jeweils 2 Kanten (Bahnränder) zu einer Leiterbahn zusammenfassen. Zudem müssten andere, im Bild vorkommenden Geraden ausgeschlossen werden.
Dieses Anpassungsverfahren scheint also relativ kompliziert zu sein, da es einfachere Wege gibt die Bahnen zu finden.

6.8 Alternativer Algorithmus zu Erkennung von Leiterbahnen auf Grundlage einer Färbung

Bohrungen sind relativ einfach zu finden, weil es sich um einigermaßen simple geometrische Objekte handelt. Da alle Leiterbahnen schließlich in einer Bohrung enden, motiviert die Idee einen Algorithmus zu schreiben, der bei bereits erkannten Bohrungen prüft, ob eine Leiterbahn in sie mündet.
Ein Beispiel für einen solchen Algorithmus wäre folgendes Verfahren:

1. Man zeichne einen Kreis $k1$, dessen Radius etwas größer ist als der Abstand vom Mittelpunkt der Bohrung bis zum äußeren Ende der umgebenden Isolierschicht.
   Da die gefundenen Bohrpunkte nicht immer perfekt in der Mitte liegen, untersucht man auch die Umgebungen jedes Bohrpunktes.
2. Falls in diesem Kreis ausgenommen von „weiß“ und „schwarz“ nur zwei Farben vorkommen, so weiß man, dass es sich entweder um eine Bohrung mit Leiterbahn handelt, oder aus Versehen eine „fremde“ Region geschnitten wurde.
3. Betrachtet man das Vorkommen der selteneren Farbe in einem Kreis $k2$, dessen Radius etwas kleiner ist, als der Radius von $k1$, so kann man anhand eines einfachen Vergleichs des Mengenverhältnisses in $k1$ und $k2$ schätzen, dass es sich um ein Leiterbahn handeln kann.
   Dies liegt daran, dass die Farbe der Leiterbahn in beiden Kreisen in ungefähr gleich vielen Pixeln auftauchen muss (im inneren Kreis evtl. etwas häufiger).

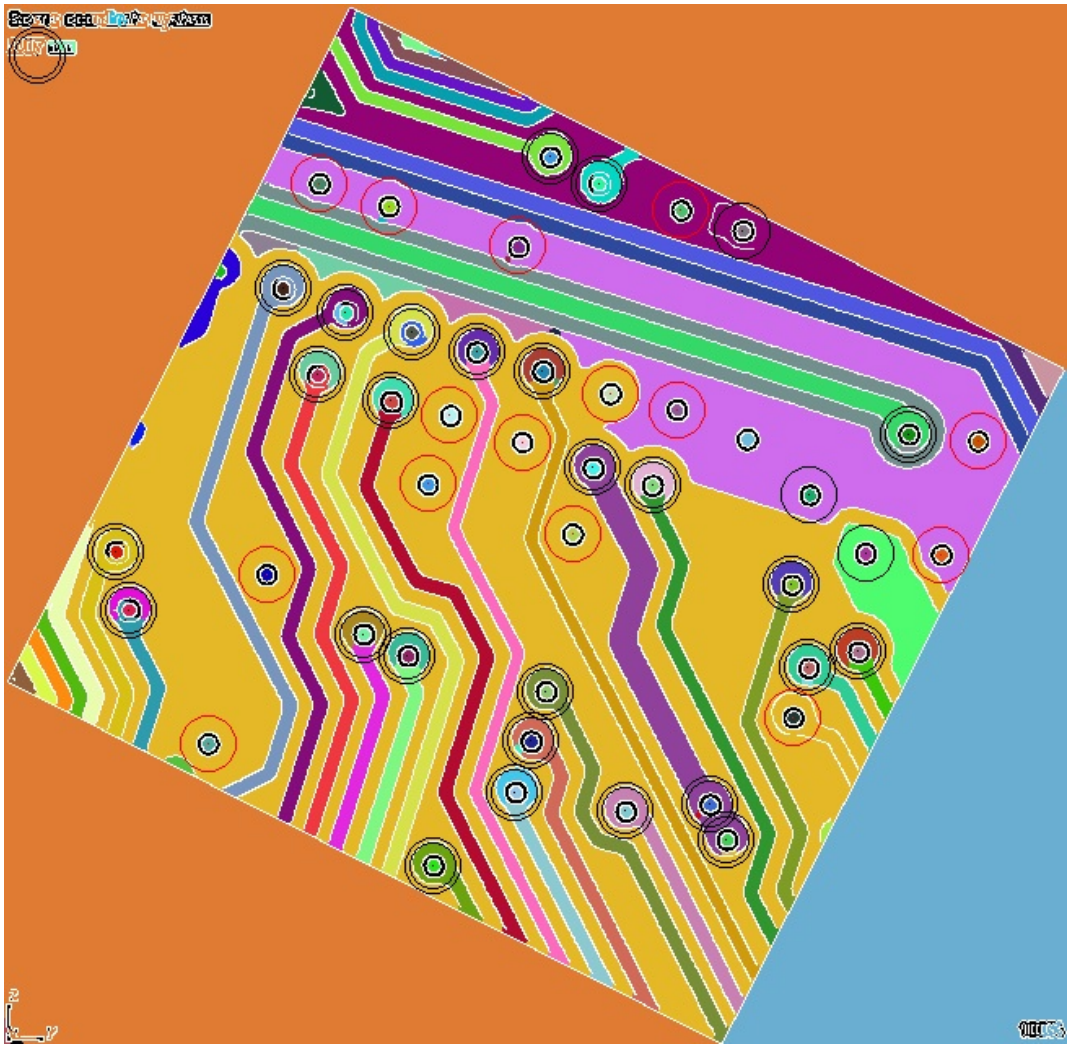

Abbildung 6.16: Eingezeichnete Kreise (bei roten Kreisen ist bereits das Kriterium aus Schritt zwei verletzt.)

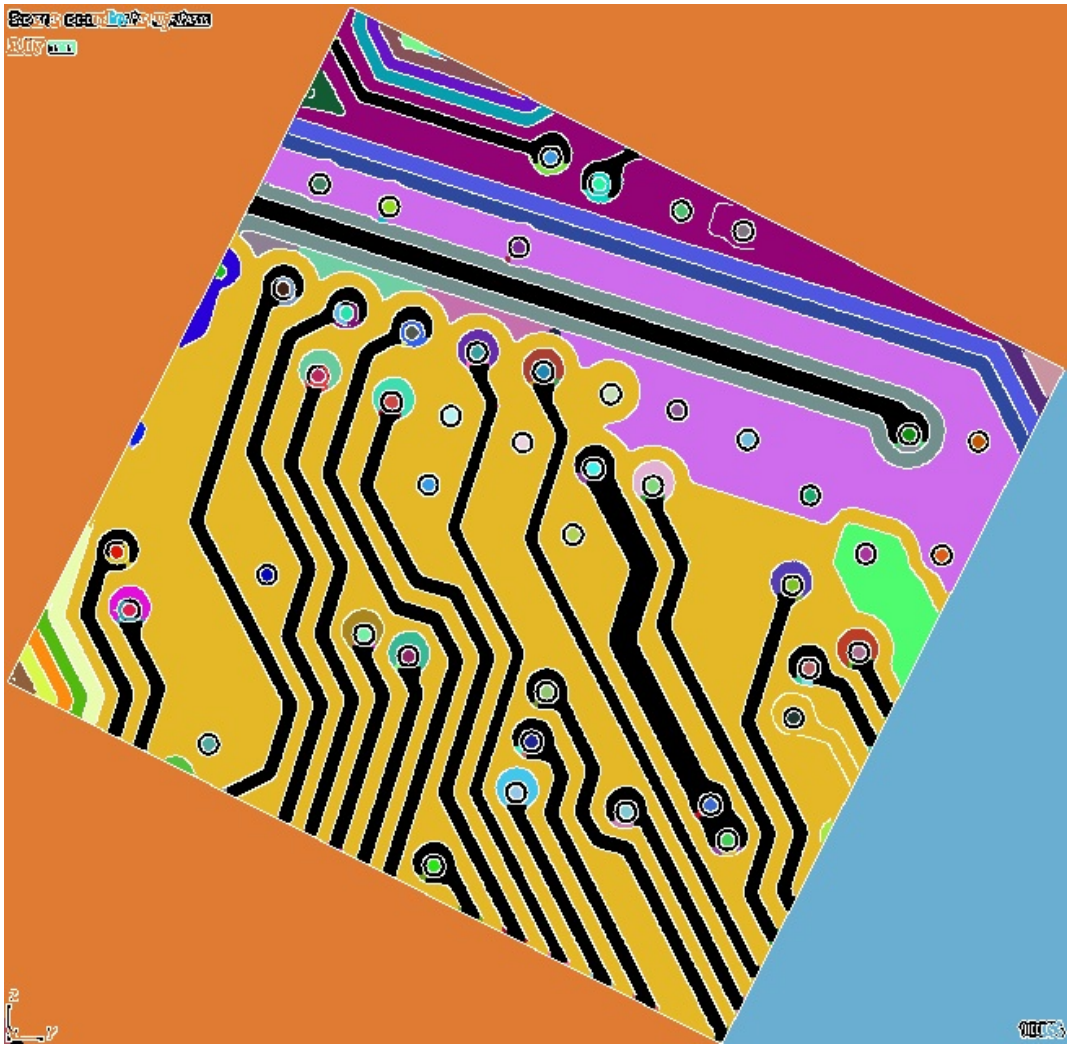

Abbildung 6.17: Ein Resultat des Algorithmus (gefundene Leiterbahnen sind schwarz markiert)

6.8.1 Bewertung:

Der Algorithmus ist relativ schnell, da er nur die bereits gegebenen Bohrungen untersucht und nicht daher nicht das komplette Bild durchforstet.
Gegenüber Fehlern im „Canny"-Bild ist der Algorihtmus nur insofern robust wie es der verwendete Färbealgorithmus ist.
Problematisch ist natürlich auch hier, dass der Algorithmus speziell auf das Problem zugeschnitten wurde und daher an Universalität einbüßt.

6.9 Weiterer alternativer Algorithmus zu Erkennung von Leiterbahnen

Da der vorherige Algorithmus von Robustheit des Färbealgorithmus abhängt und daher im Allgemeinen anfällig ist für Fehler des Canny-Edge-Detektors, wäre ein Algorithmus interessant, welcher ohne Färbung auskommt.
Da die Bohrungen bisher zuverlässig erkannt wurden und es das Verfahren extrem beschleunigt, soll auch hier wieder von den gefundenen Bohrungen ausgegangen werden.
Der hier vorgeschlagene, auf das Problem abgestimmte Algorithmus sucht im Wesentlichen die Kante(n) einer eventuellen Leiterbahn und untersucht ihr Verhalten bezüglich der Umrundung des Bohrpunktes:

1. Man zeichne wieder einen Kreis, dessen Radius etwas größer ist als der Abstand vom Mittelpunkt der Bohrung bis zum äußeren Ende der umgebenden Isolierschicht.
   Da die gefundenen Bohrpunkte nicht immer perfekt in der Mitte liegen, untersucht man auch die Umgebungen jedes Bohrpunktes.
2. Falls es auf dem Kreis genau zwei weiße Cluster gibt und deren euklidischer Abstand im Toleranzbereich der Breite einer Leiterbahn liegt, so kann man sagen, dass es sich entweder um die beiden Seiten der Leiterbahn handelt oder keine Leiterbahn von der Bohrung ausgeht und zufällig zwei fremde Punkte mit dem richtigen Abstand gefunden wurden.
3. Um letzteren Fall mit hoher Wahrscheinlichkeit auszuschließen könnte man beispielsweise jeweils der zugehörigen weißen Linie eines Clusters folgen und schließlich untersuchen, ob die Vereinigung der beiden Linien den Bohrpunkt umrundet (bspw. Quadrantencheck: Liegen in jedem Quadranten um den Bohrpunkt min. x Pixel?).

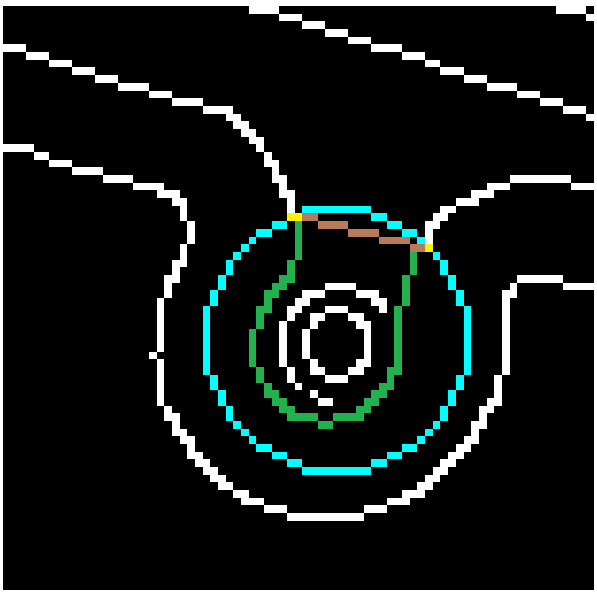

Abbildung 6.18: Beispiel zu Verdeutlichung: Der in Schritt 1 gezeichnete Kreis ist blau; Die gefundenen Cluster sind gelb; Der gemessene Abstand ist braun; Die Vereinigung der beiden Linien grün.

Um nun die tatsächliche Leiterbahn zu extrahieren könnte man beispielsweise den Floodfill Algorithmus auf der braunen Linie im Bild starten und somit alle Pixel der Leiterbahn finden. Das würde jedoch dem Vorhaben widersprechen, ohne Färbung und der damit verbunden, bereits angesprochenen Schwäche auszukommen.
Daher bietet es sich beispielsweise folgendes Verfahren zur Extraktion der Leiterbahnen an (Vorraussetzung: Die Leiterbahnen machen keine scharfen Kurven):

1. Man schießt einen Strahl, ausgehend vom Bohrpunkt in Richtung Leiterbahn, dh. durch die Mitte der gezeichneten, braunen Linie.
2. Der Strahl wird am Punkt p gestoppt, und zwar n Pixel vor einer weißen Linie.
3. Anschließend vergleicht man n mit n_{links} und n_{rechts}, welche entstehen, wenn man, anstatt von p geradeaus zu gehen um 45° nach links bzw. nach rechts geht.
4. Den längsten der drei Wege wählt man als neuen Startweg, halbiert den Winkel und macht weiter bei Schritt 3. Durch diesen Trick wird bei den Leiterbahnen für jeden aktuellen Punkt p der längste Weg in die richtige Richtung gefunden (siehe nachfolgende Grafik).
5. Wenn man den längsten Strahl ausgehend vom Punkt p gefunden hat, dann wird p neu gesetzt, indem man mit diesem Strahl wieder bei Schritt 2 weitermacht. Falls kein neuer Strahl gefunden wurde ist man fertig.

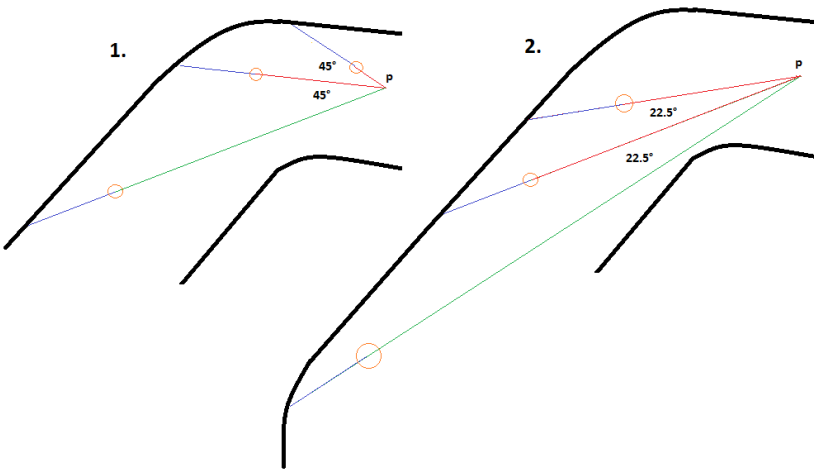

Abbildung 6.19: Zwei Iterationsschritte zu Verdeutlichung: Der grüne Weg ist jeweils der längste und wird daher als Ausgangsrichtung für die nächste Winkelhalbierung gesetzt.

Dieses Vorgehen führt dazu, kleinere Lücken im Kantenbild mit einigermaßen hoher Wahrscheinlichkeit übersprungen werden, insbesondere dann, wenn sie auf einem langen, geradlinigen Abschnitt auftauchen. Dieser Vorteil ist im folgenden Bild zu sehen, in welchem auch die Leiterbahn gefunden wurde, welche beim vorherigen Verfahren aufgrund der fehlerhaften Färbung nicht erkannt wurde.

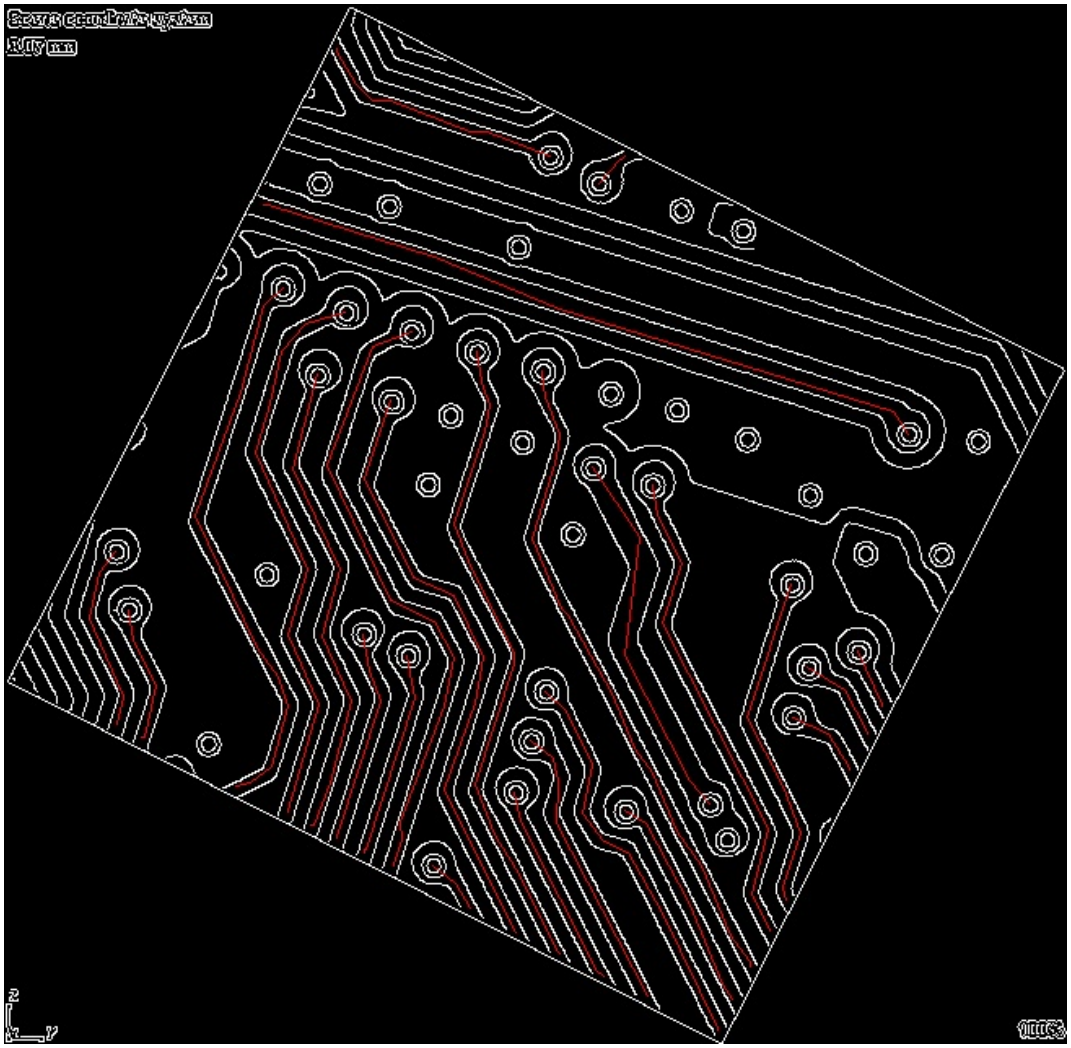

Abbildung 6.20: Ein Resultat des Algorithmus (gefundene Leiterbahnen sind rot markiert). Es wurden zusätzlich Leiterbahnen, welche zwischen zwei Bohrungen verlaufen kombiniert.

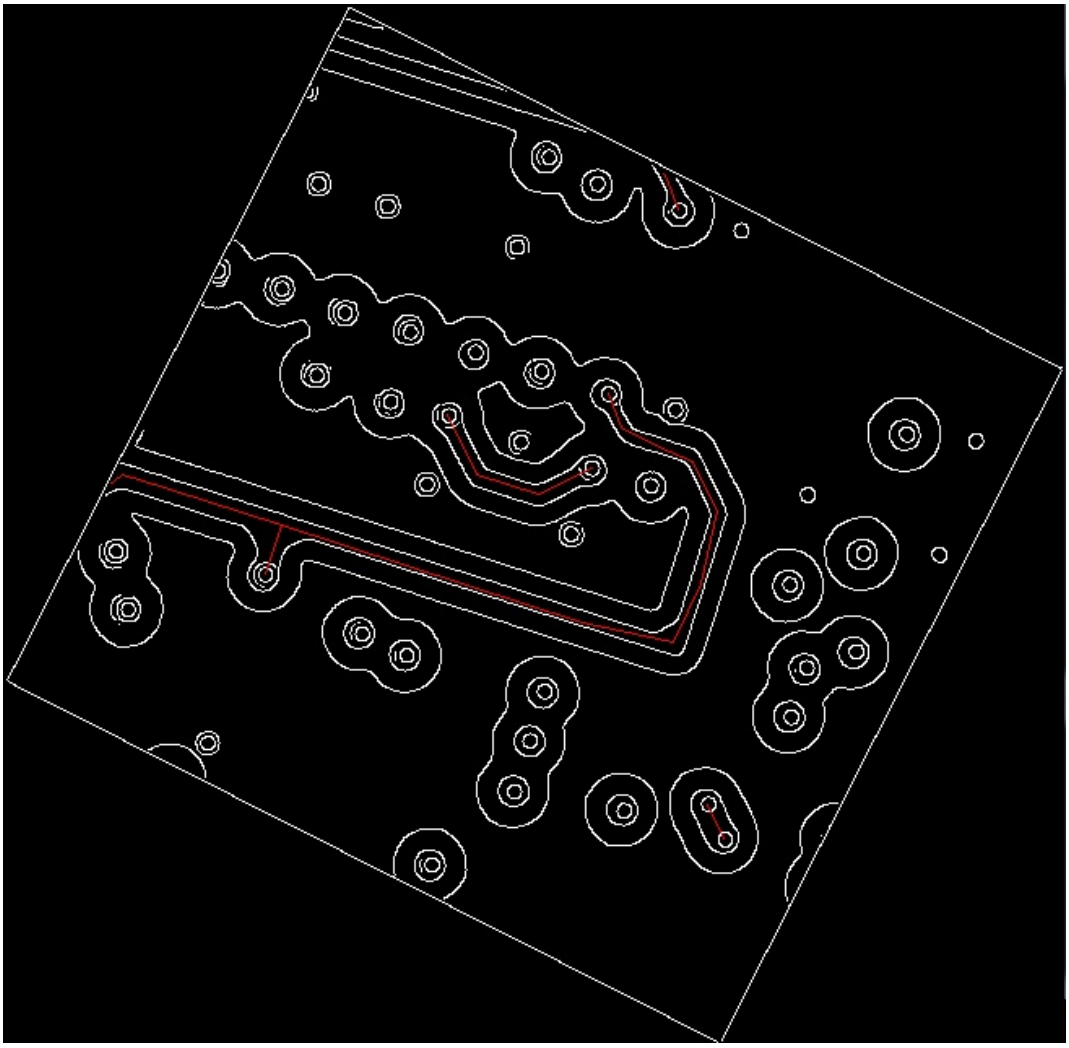

Abbildung 6.21: Ein anderes Resultat des Algorithmus (gefundene Leiterbahnen sind rot markiert). Außerdem wurden Leiterbahnen, welche zwischen n Bohrungen verlaufen zu einem Graph kombiniert.

6.9.1 Bewertung:

Durch die beidseitige Verfolgung der von den Clustern ausgehenden Linien und dem anschließenden Check deren Vereinigung bezüglich dem Quadrantenkriterium hat sich das Verfahren für die Erkennung von Leiterbahnen auch bei Lücken im Canny-Edge-Bild bewährt.
Das Verfahren zur nachfolgenden Extraktion der Leiterbahnen hat in den getesteten Szenarien gut funktioniert und ist einigermaßen einfach zu implementieren.
Wenn ein Strahl jedoch zufällig durch eine Lücke im Kantenbild springt, dann hat das unab-

sehbare Konsequenzen und die Leiterbahn muss aus der Liste gestrichen werden. Ersteres ist wiederum einem Algorithmus nicht auf trivialem Wege ersichtlich und diesbezüglich müssen weitere Anstrengungen unternommen werden.
Auch dieser Algorithmus ist extrem schnell und dürfte keinerlei Performanz-Probleme bereiten.

7 3D - Verfahren

7.1 RANSAC

RANSAC (**RA**ndom **SA**mple **C**onsensus) [RCB81] ist ein iteratives Verfahren zur Schätzung eines mathematischen Models anhand von Beobachtungsdaten mit Ausreißern. Der Algorithmus ist nicht-deterministisch, da ihm probabilistische Ansätze zu Grund liegen. Aufgrund seiner Robustheit wird er häufig im Bereich des maschinellen Sehens eingesetzt.

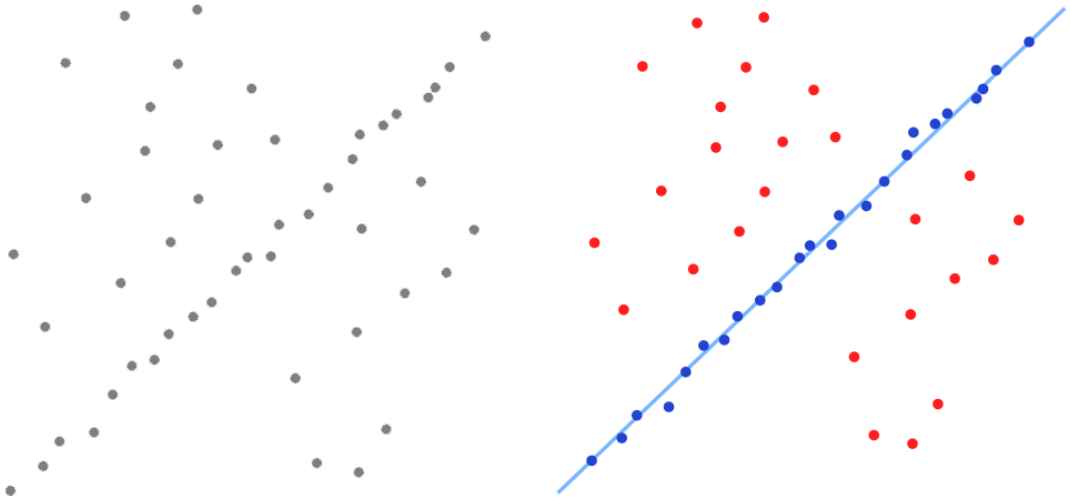

Abbildung 7.1: In einen Datensatz mit vielen Ausreißern wird eine Linie eingepasst.

7.1.1 Der Algorithmus

Für das zu erkennende Objekt wird ein parameterabhäniges Modell erstellt. Danach werden die Daten iterativ auf mögliche Vorkommnisse eines auf das Modell passenden Objekts getestet, dafür werden iterativ zufällige Punkte aus den Daten selektiert und als hypothetische Einlieger des Objekts betrachtet und das Modell an diese Punkte angepasst. Für eine Linie wären dies zwei Punkte um sie ausreichend zu beschreiben. Nun wird das Modell getestet:

1. Alle anderen Punkte werden gegen das Modell getestet und falls sie dazu passen ebenfalls als mögliche Einlieger gespeichert.
2. Das Modell wird akzeptiert wenn genug Einlieger gefunden werden.

3. Das Modell auf Basis aller gefundenen Einlieger neu berechnet und evaluiert.

Nach ausreichend vielen Iterationen wird das beste Modell benutzt um alle Punkte des Objekts zu identifizieren. Anschließend wird das gefundene Objekt aus den Daten gelöscht und gegebenenfalls nach weiteren Vorkommnissen gesucht.

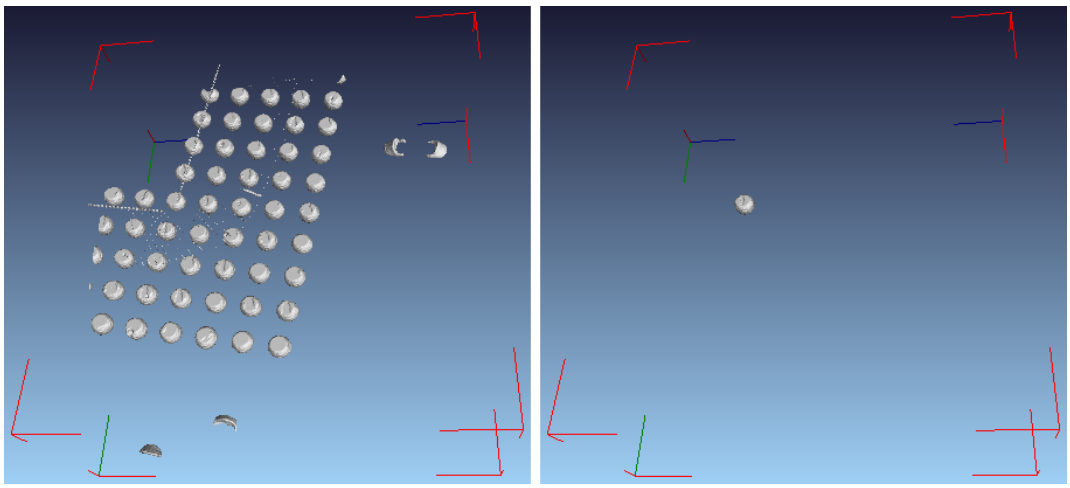

Abbildung 7.2: In einem per Threshhold-Verfahren vorverarbeiteten Datensatz wird anhand eines einfachen Modells eine Kugel identifiziert.

7.1.2 Bewertung

Der Algorithmus eignet sich sehr gut um in großen Datenmengen schnell Instanzen von Modellen zu finden. Für den diskutierten Anwendungsfall ist er leider vergleichsweise langsam. Der Grund dafür ist, dass die zu suchenden Objekte schon durch einen Dichte-Threshhold sehr gut vor-isoliert werden können. Man kann sich deshalb direkt darauf konzentrieren einzelne Punktmengen auf bestimmte Eigenschaften hin zu untersuchen. Der große Vorteil von RANSAC schnell mögliche Kandidaten zu entdecken wird deshalb negiert.

7.2 Alternativer Algorithmus zur Erkennung von Lötkugeln und Bonddrähten

Der wohl am naheliegenste Ansatz Objekte in einem Datensatz zu finden, ist den Datensatz in einem hinreichend kleinen Raster abzusuchen und an jedem Punkt das damit verbundene Objekt auf die Ähnlichkeit mit den gesuchten Objekten hin zu untersuchen. Ein solcher Ansatz ist insbesonere bei den großen und einigermaßen wohlgeformten Lötkugeln sehr intuitiv, wie in der folgenden Abbildung zu sehen ist.

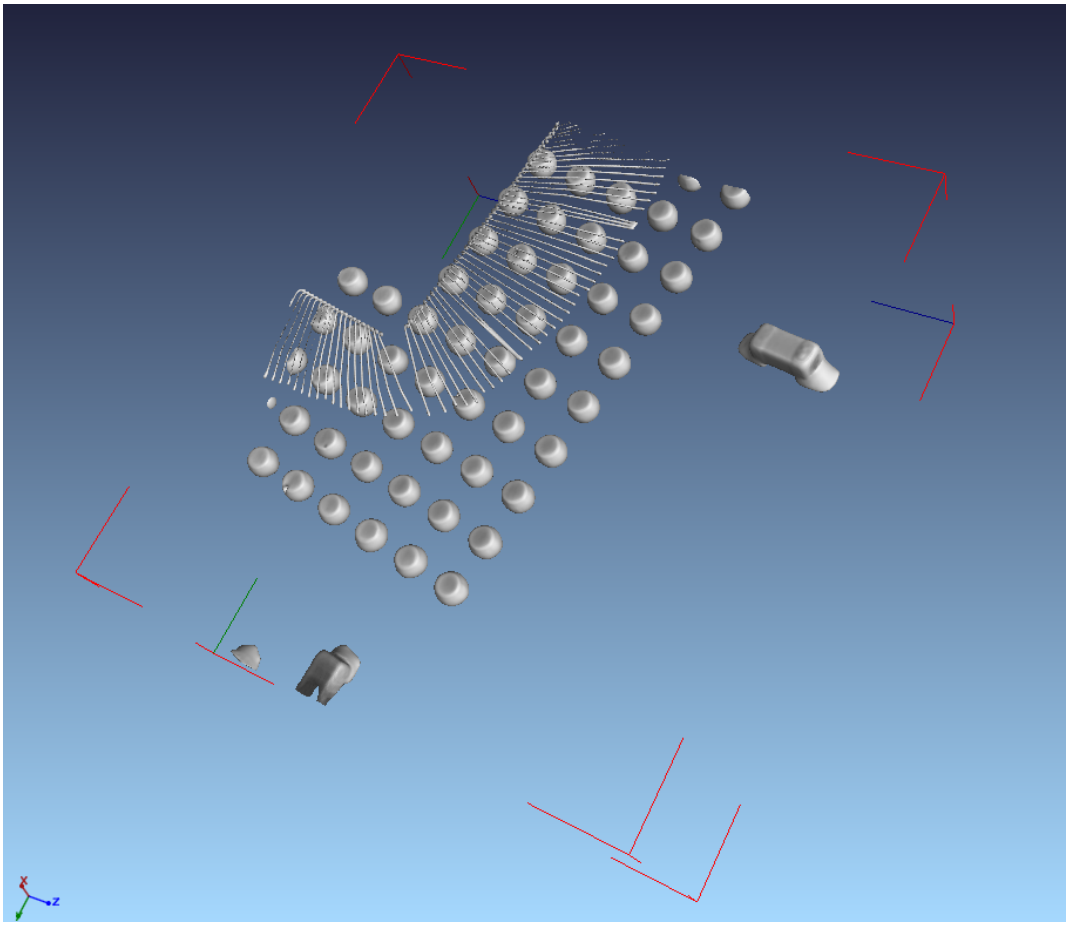

Abbildung 7.3: Die gesuchten Kugels sind hier ab einem bestimmten Reflektions-Schwellwert sehr einfach zu identifizieren, die Bonddrähte sind dagegen eher schwierig unter einen Hut zu bringen.

Das Vorgehen bei einem solchen Algorithmus zur Suche der gezeigten Lötkugeln könnte also im einfachen Fall wie folgt aussehen:

procedure FINDSPHERES(int *radius*, 3DUniformVoxelgrid g)
  $searchStep = int(\sqrt{2} \cdot radius)$
  for all Voxel $v \in g$ **by Step** $searchStep$ **do** // Kurzform für die 3 Schleifen.
    $value = getValue(v)$
    if $value > 40000$ **then**
      if $v \notin investigated$ **then**
        $res = getFloodfill3D(v, value, tolerance = 8000)$
        $investigated = investigated \cup res$
        if $isApproxSpheric(res.min, res.max)$ **then** $found = found \cup res$
      end if
    end if
  end for
end procedure

Hierbei sei als einfache Approximation nachfolgender Check gegeben (ggf. sollten hier Ansätze gewählt werden, die etwas durchdachter sind):

procedure ISAPPROXSPHERIC(int *min*, int *max*)
  if $max[1] \vee max[2] \vee max[3] = 0$ **then return** False
  $xy = 0.65 < |min[0] - max[0]|/|min[1] - max[1]| < 1.35$
  $xz = 0.65 < |min[0] - max[0]|/|min[2] - max[2]| < 1.35$
  $zy = 0.65 < |min[2] - max[2]|/|min[1] - max[1]| < 1.35$
  return $xy \wedge xz \wedge zy$ // „bounding box“ ist näherungsweise kubisch.
end procedure

Für die Suche nach Bonddrähten reicht es, im obigen Algorithmus die Schrittweite, den Schwellwert und den Approximationscheck zu ersetzen. Letzterer könnte im einfachen Fall beispielsweise wie folgt aussehen:

procedure ISLONGENOUGH(int *min*, int *max*)
  return $40 < \sqrt{|min[0] - max[0]|^2 + |min[1] - max[1]|^2 + |min[2] - max[2]|^2}$
end procedure

Wie in nachfolgender Abbildung zu sehen ist, kann selbst ein derart einfacher Test reichen um die, durch den Schwellwert ohnehin schon gefilterten Objekte, perfekt einzuschränken.

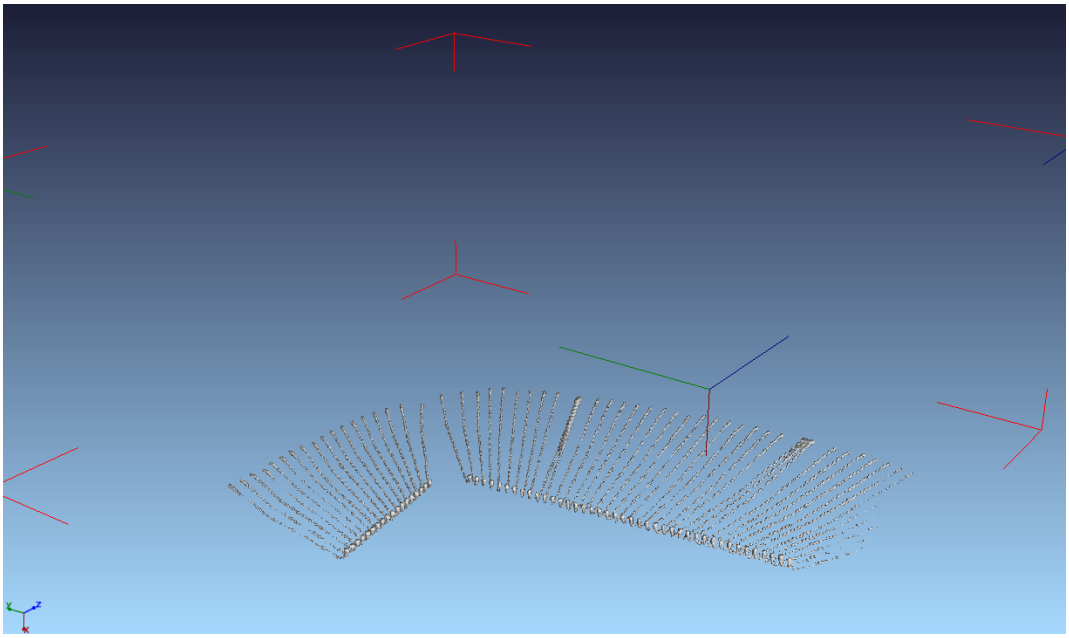

Abbildung 7.4: Der Algorithmus erkennt im Beispieldatensatz alle Bonddrähte und Kugeln (hier nicht eingezeichnet). Durch die automatische Interpolation des Viewers sind die dargestellten Drähte etwas verwaschen.

7.2.1 Bewertung

Das Vorgehen des Algorithmus ist sehr intuitiv und einfach. Bei allen Tests reichten außerdem unkomplizierte Approximationsbedingungen, um die gesuchten und ausschließlich die gesuchten Objekte zu extrahieren. Besonders einfach zu beschreibende geometrische Objekte, wie beispielsweise Kugeln, sind sehr leicht zu finden.
Um den Algorithmus auf ein Objekt zu eichen muss lediglich die Schrittweite und der Schwellwert angepasst werden und das vom Floodfill-Algorithmus zurückgegebene Objekt in den abschließenden Bedingungen hinreichend genau beschrieben werden.
Ein Datensatz kann bei räumlich ausgedehnten Objekten schnell den Datensatz durchsuchen, da bei weitem nicht jeder Voxel untersucht werden muss (Bei Kugeln reicht beispielsweise eine Schrittweise von $\lfloor \sqrt{2} \cdot r \rfloor$). Die Geschwindigkeit ist selbstverständlich abhängig von der Größe und dem Vorkommen der gesuchten (und ähnlicher) Objekte.

8 Zusammenfassung und Empfehlung

In dieser Fachstudie wurden Verfahren zur Erkennung von Objekten in CT-Scans planarer Hochfrequenzschaltungen untersucht. Da die zu suchenden Objekten begrenzt und sehr unterschiedlich waren sind auch die behandelten Algorithmen in Funktionsweise und Einsatzgebiet stark verschieden. Manche der algorithmischen Ansätze wurden auch spezifisch für die Erkennung einer Art von Objekt konzipiert. Deshalb macht es kaum Sinn einen globalen Vergleich durchzuführen und wir haben uns entschieden unterteilt nach den Objektgruppen jeweils ein einzelnes Fazit zu ziehen.

8.1 Bohrpunkte

Zur Erkennung von Bohrpunkten sind 3d-Verfahren grundsätzlich geeignet, wobei nun eben nach Zylindern gesucht werden muss. Es wurden jedoch keine Verfahren untersucht, die einen solchen Ansatz implementieren. Stattdessen wurden hier ausschließlich 2d-Verfahren eingesetzt und bewertet.
Das SURF Verfahren als Variante des SIFT erzielte gute Resultat und erkannte alle Bohrungen. Jedoch wurden einige Merkmalsvektoren irrtümlich als Bohrung identifiziert.
Sowohl das untersuchte Verfahren zum Template-Matching als auch die Houghtransformation hatten das Problem, dass die lokalen Minima nach der Faltung sich zueinander unterschiedlich stark ausgeprägt hatten. Das eingesetzte Schwellwert-Verfahren lieferte bei der Houghtransformation jedoch wesentlich bessere Resultate.
Am erfolgreichsten stellte sich jedoch der selbst implementierte, auf das Problem zugeschnittene Algorithmus zur Erkennung von Kreisen in Kantenbildern heraus.
Er erkannte ohne Fehler alle Bohrpunkte in allen getesteten Bildern, was selbstverständlich auch an den fast fehlerfreien Resultaten des Canny-Edge-Detektors liegt.
Dabei handelt es sich zusätzlich um den (mit Abstand) schnellsten Algorithmus, da er ohne Faltung auskommt.

8.2 Leiterbahnen

Da Leiterbahnen einen zweidimensionalen Verlauf besitzen boten die dreidimensionalen Verfahren keinen nennenswerten Vorteil, weil keine zusätzlichen Informationen verarbeitet werden konnten. Deswegen wurden für die Erkennung von Leiterbahnen auch keine dreidimensionalen Verfahren behandelt, sondern ausschließlich auf 2D-Slices gearbeitet.
Das zuerst untersuchte Verfahren war die Houghtransformation für Linien. Es stellte sich

allerdings schnell heraus, dass die Geradenerkennung nur bedingt für die Erfassung von Leiterbahnen geeignet war. Der Grund dafür war der unregelmäßige Verlauf der Leiterbahnen mit vielen Kurven und Mündungen in Bohrpunkten, welches zu Unmengen unverknüpfter Graden führte.
Da eine Leiterbahn zwar aus einer begrenzten Zahl von Bauteilen besteht, diese aber quasi frei kombiniert sein können, eigneten sich SIFT/SURF und Template-Matching selbstverständlich auch nicht zur Erkennung. Für diese Verfahren müsste das zu erkennende Objekt immer die gleichen äußerlichen Eigenschaften aufweisen.
Die zwei alternativen Algorithmen erwiesen sich als ausgezeichnet bei der Leiterbahnerkennung. Durch die Ausnutzung objektspezifischer Eigenschaften, wie der Mündung in Bohrpunkten, und der meist hohen Qualität der Kantenerkennung konnten quasi perfekte Erkennungsraten erreicht werden. Der zweite Algorithmus verbesserte dabei das schon sehr gute Ergebnis der ersten Alternative noch einmal. Die Abhängigkeit von FloodFill wurde eleminiert und gleichzeitig die Anfälligkeit gegenüber Fehlern im Canny-Bild weiter gesenkt. Der zweiter Alternativ-Algorithmus stellt damit die beste gefundene Lösung für das Problem dar und ist auch einer der performantesten der getesteten Algorithmen.

8.3 Lötkugeln

Bei der Erkennung der Lötkugeln stellte sich erneut die Frage, ob nur eine Klasse von Verfahren behandelt werden sollte. Durch ihren hohen Reflektanzwert sind die Lötkugeln sowohl auf den Slices als auch in den Voxeldaten leicht zu isolieren. Der größte Vorteil der dreidimensionalen Algorithmen ist, dass mit dem Objekt als Ganzes gearbeitet werden kann. Arbeitet man hingegen nur mit den Slices, so müssen die erkannten Teil-Kreise erst wieder zu einem Ganzen zusammengeführt werden. Dadurch stellen sich zahlreiche neue Probleme, die in ihrer Menge schlussendlich dazu führten, dass ausschließlich dreidimensionale Verfahren getestet wurden.
Eines der bekanntesten Verfahren zur Erkennung von Objekten in verrauschten Daten ist der RANSAC-Algorithmus. Da sich für eine Kugel sehr leicht ein Modell erstellen lässt, konnten mit RANSAC zuverlässig Lötkugeln gefunden werden. Leider relativierte die Möglichkeit die Kugeln durch Thresholds vorzuisolieren den Geschwindigkeitsvorteil, den RANSAC durch die schnelle Untersuchung möglicher Kandidaten erzielt. Insgesamt war die Perfomanz leider enttäuschend.
Der intuitive Algorithmus, welcher den Datensatz in einem hinreichend engen Raster durchsucht und getroffene Objekte auf ihre Kugelförmigkeit überprüft, scheint einen guten Ansatz darzustellen, solange die Akzeptanzbedingung fein genug definiert wird.
Der Algorithmus ist extrem schnell und erkannte alle gesuchten Objekte, da es sich bei diesen um sehr einfache geometrische Objekte handelt, die hier leicht abgegrenzt werden können.

8.4 Bonddrähte

Den Bonddrähten, als schwierigste zu erkennende Objekte, wurde insgesamt eher wenig Zeit gewidmet, da kein Algorithmus wirklich geeignet erschien.
Die Effektivität zweidimensionaler Verfahren wurde gar nicht erst untersucht, da es spontan äußerst schwierig erschien entsprechende Slices oder Projektionen zu extrahieren.
Der bei Kugeln eingesetzte RANSAC-Algorithmus ist theoretisch natürlich geeignet, jedoch könnte es sich aufgrund der Vielfalt der Drähte als problematisch herausstellen ein geeignetes Modell zu konstruieren.
Der einzige Algorithmus der tatsächlich zur Erkennung von Bonddrähten eingesetzt wurde, war der intuitive, das Bild rasterartig durchsuchende Algorithmus auf Basis des dreidimensionalen Floodfill.
Dieser erkannte alle Bonddrähte die lang genug waren und bereitete im Test keine Performanzprobleme. Dennoch ist seine universelle Einsatzfähigkeit eher fraglich, zumindest ohne die Akzeptanzbedingung zu verfeinern.

Literaturverzeichnis

[BFRR95] T. Braunl, S. Feyrer, W. Rapf, M. Reinhardt. *Parallele Bildverarbeitung*. Addison-Wesley, 1995. (Zitiert auf Seite 19)

[BTG05] H. Bay, T. Tuytelaars, L. V. Gool. SURF: Speeded Up Robust Features. 2005. (Zitiert auf Seite 36)

[Can86] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, 1986. (Zitiert auf Seite 15)

[Har09] P. E. Hart. How the Hough transform was invented. *IEEE Signal Processing Magazine*, 2009. (Zitiert auf Seite 21)

[Kru94] A. Kruger. Median-cut color quantization. *Dr. Dobbs Journal*, S. 46–54 und 91–92, 1994. (Zitiert auf Seite 18)

[Low04] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. (Zitiert auf Seite 33)

[RCB81] M. A. F. und Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. 1981. (Zitiert auf Seite 59)

Alle URLs wurden zuletzt am 31.05.2012 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Marcus Hörger, Andreas Paul, Felix Reeh)