Universität Würzburg
Institut für Informatik
Research Report Series

# Einsatz von topologieerhaltenden neuronalen Netzen zur Interpretation von Leiterplattendaten

W. Jodl*, M. Heuler*, U. Rothaug** und
K. Leibnitz*

Report Nr.: 206 Juli 1998

* Lehrstuhl für Informatik III, Universität Würzburg
Am Hubland, 97074 Würzburg
Tel: 0931 8885513, Fax: 0931 8884601
E-Mail: jodl@informatik.uni-wuerzburg.de

** ATG Test Systems GmbH
Zum Schlag 3, 97877 Wertheim-Reicholzheim
Tel: 09342 2910, Fax: 09342 39510
E-Mail: atg.ceo@t-online.de

# Einsatz von topologieerhaltenden neuronalen Netzen zur Interpretation von Leiterplattendaten

**Wolfgang Jodl, Marius Heuler, Kenji Leibnitz**
Lehrstuhl für Informatik III, Universität Würzburg, Am Hubland, 97074 Würzburg.
Tel: 0931-8885513, Fax: 0931-8884601, E-Mail: jodl@informatik.uni-wuerzburg.de

**Uwe Rothaug**
ATG Test Systems GmbH, Zum Schlag 3, 97877 Wertheim-Reicholzheim.
Tel: 09342-2910, Fax: 09342-39510, E-Mail: atg.ceo@t-online.de

**Zusammenfassung:**
In dieser Studie wird ein neuer Algorithmus vorgestellt, der in der Lage ist, die Zusammenhänge in der Struktur bei unbestückten Leiterplatten zu erkennen. Fehler, die aufgrund unsauberer Verarbeitung entstanden sind, können erkannt und behoben werden. Der verwendete Ansatz basiert auf wachsenden topologieerhaltenden Karten mit dynamischer Netzstruktur. Dieser Bericht beschreibt das neuronale Verfahren und die Anwendung auf reale Leiterplatten.

## 1 Einleitung

Reparatur von defekten, noch unbestückten Leiterplatten ist ein Arbeitsschritt, der von Leiterplattenherstellern wie z.B. großen Computerfirmen, bisher sehr wenig betrieben wurde. Der geringe Wert solcher unbestückter Leiterplatten, bis zu 100 DM, im Vergleich zum hohen Verkaufspreis einer bestückten Leiterplatte (Motherboards für Personal Computer wurden mit Preisen bis 7.000.- DM verkauft), resultierte in folgender Strategie der Leiterplattenhersteller. Unbestückte Leiterplatten, im folgenden auch *Bare Boards* genannt werden zwar elektrisch auf Kurzschlüsse zwischen verschiedenen Leiterbahnen und Unterbrechungen einer Leiterbahn getestet, sobald aber ein Fehler gefunden wird, ist diese Leiterplatte im weiteren Produktionsprozeß, der Bestückung mit Bauteilen, nicht mehr zu verwenden.
Zwei Faktoren zwangen Produzenten von Bare Boards dazu, die Reparatur unbestückter Leiterplatten als Prozeßschritt einzuführen:

1. Beeinflußt durch den rapiden Preisverfall in der Elektronik- und insbesondere in der Computerbranche waren Leiterplattenhersteller gezwungen, selbst relativ kostengünstige Leiterplatten zu reparieren, sobald eine Neuproduktion im Vergleich teurer ist.
2. Durch die Miniaturisierung der Elektronik mit gleichzeitiger Verbesserung der Leitungsfähigkeit wurden und werden Bare Boards immer komplexer und damit auch erheblich teurer. Solche Leiterplatten werden immer häufiger in der Telekommunikation (z.B. Handys), der Autoindustrie (z.B. elektronische Fahrwerksteuerungen) und im Haushalt (bei nahezu allen Geräten) eingesetzt. Teuere Leiterplatten werden häufiger repariert, da eine Neuproduktion erhebliche Kosten verursacht.

Die Reparatur von Leiterplatten ist ein Prozeßschritt, der für alle Produzenten von Leiterplatten zwingend geworden ist. Er beinhaltet aber ein wesentliches technisches Problem, das im nächsten Abschnitt ausführlich erläutert werden soll.

## 1.1 Technischer Hintergrund und Problembeschreibung

Da unbestückte Leiterplatten bisher nur auf „gut“ oder „schlecht“ getestet wurden, war es unnötig, den Fehler zu lokalisieren. Diese Information, die für die Reparatur Voraussetzung ist, wurde nicht benötigt.
Aus diesem Grund benutzt der elektrische Leiterplattentest nicht den Verlauf der Leiterbahnen als Eingabedaten, sondern als Kontaktierpunkte für die elektrische Messung werden *Bohrungen* oder *SMDs* (Surface Mounted Device) verwendet. Dies wurde so realisiert, um mit einer wesentlich geringeren Datenbasis arbeiten zu können. CAD-Daten, die den Verlauf der Leiterbahnen darstellen, stellen bei Standard-Leiterplatten Datenmengen bis zu 100 MB dar. Eine solche Informationsmenge ist auf den Testsystemen, die in Echtzeit arbeiten und testen müssen, mit vertretbarem Aufwand nicht mehr schnell genug zu verarbeiten.
Demzufolge kann als Ergebnis des elektrischen Tests in den meisten Fällen nicht der tatsächliche Fehlerort angegeben werden. Wird zum Beispiel ein Kurzschluß erkannt, so wird als Resultat des elektrischen Tests jeweils ein repräsentativer Punkt der zwei Netze, die fälschlicherweise miteinander verbunden sind, dem Benutzer als Fehlerort gemeldet. Es werden diejenigen Punkte genommen, die in der zweidimensionalen Ebene den geringsten Abstand haben. Der tatsächliche Fehlerort ist damit jedoch normalerweise nicht gefunden. Der Fehler ist z.B. dadurch entstanden, daß durch einen Fehler in der Fertigung zwei Leiterbahnen sich berühren (siehe Abb. 1).

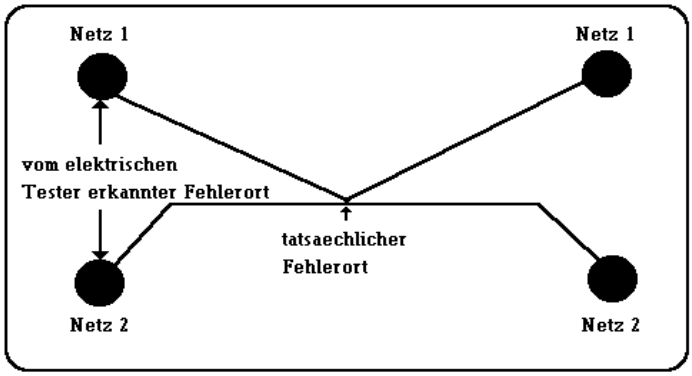


***Abb. 1: Schematisierte Darstellung eines Kurzschluß auf einer Leiterplatte***

Abb. 1 soll in stark vereinfachter Form die Problematik verdeutlichen. Bei sehr großen, fehlerhaften Netzen, oder wenn die Bohrpunkte sehr weit auseinander liegen, müssen die zum jeweiligen Netz gehörenden Leiterbahnzüge auf der Platine gesucht werden.
Da jedoch die meisten *PCB* (Printed Circuit Boards) aus mehreren Lagen bestehen, ist das Nachverfolgen der Leiterbahnen mit dem Auge nahezu unmöglich.

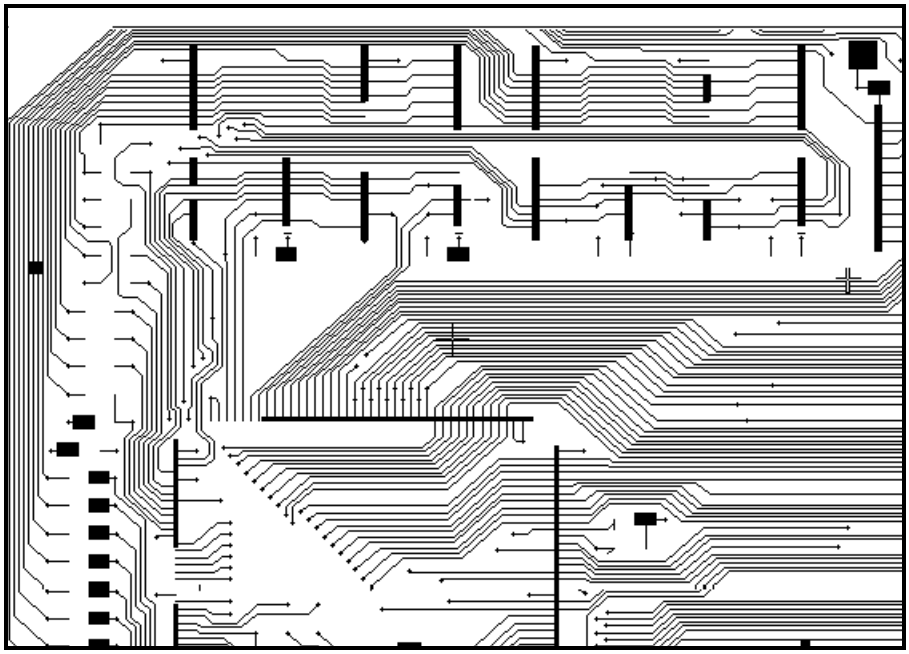

***Abb. 2: Bildausschnitt einer komplexen Leiterplatte***

Aus diesem Grund existieren Programme mit einer graphischen Benutzeroberfläche, die dem Benutzer ermöglichen, auf dem Bildschirm eine Projektion des Boards zu betrachten, sowie die zugehörigen Fehler zu erkennen.
Diese Programme haben als Eingabedaten das Layout der Leiterplatte. Leider hat sich bisher weltweit kein Standard für CAD-Datenformate etabliert. Als einziger Pseudostandard hat sich das sogenannte *Gerber-Format* durchgesetzt. Gerber ist eine US-Firma, die als eine der ersten weltweit seit ca. 25 Jahren (Photo-)Plotter herstellt und ihr Datenformat als Standard für alle Plotmaschinen durchgesetzt hat. Der Photoprozeß ist bei der Produktion jeder Leiterplatte notwendig. Aus diesem Grund können praktisch alle Designprogramme für Leiterplatten Daten im Gerber-Format manipulieren.
Durch die Konvertierung von beliebigen CAD- zu Gerber-Daten können erhebliche Rundungsfehler entstehen. Ein weitaus größeres Problem ist jedoch, daß das Gerber-Format ein rein auf die Wegoptimierung der Arbeitsköpfe des Photoplotters ausgelegtes Datenformat ist, aus dem sich die zum elektrischen Test und zur Reparatur von Leiterplatten notwendigen zusammenhängenden Leiterbahnnetze nicht direkt erkennen lassen.
Entscheidende Aufgabe des im Rahmen der Arbeit zu entwickelnden Algorithmus ist es deshalb, aus den Gerber-Daten, die aus einer beliebigen Anordnung von Leiterbahnstücken bestehen, elektrisch zusammenhängende Leiterbahnzüge zu ermitteln, und diese dann in *ATF* (ATG Testdaten Format) zu konvertieren.
Der Algorithmus muß ebenfalls die im Gerber-Format enthaltenen Unsauberkeiten, wie z.B. Überlappungen von Leiterbahnzügen oder Lücken in elektrisch verbundenen Leiterbahnzügen, die bei der Konvertierung von CAD- in Gerber-Datenformat entstehen, beseitigen.
Ein weiteres großes Problem bei Gerber-Daten sind sogenannte *gezeichnete* SMDs. Für die Erzeugung der Filme für die Leiterplattenproduktion mit einem Photoplotter reicht es aus, SMD Flächen aus einzelnen sich überlappenden Strichen zu zeichnen. Für den elektrischen Test ist es aber notwendig, diese gezeichnete SMD-Fläche durch eine *geflashte* SMD-Fläche (definiert als rechteckige Blende mit Länge und Breite) zu ersetzen, da auf ein SMD im Gegensatz zu Leiterbahnen ein elektrischer Testpunkt gesetzt werden muß.

# 2 Einsatz des neuronalen Algorithmus

## 2.1 Der neuronale Lösungsansatz

Zur Lösung der in Abschnitt 1.1 beschriebenen Problemstellung wurde der Einsatz eines Algorithmus, basierend auf *selbstorganisierenden Merkmalskarten*, gewählt. Solche Karten (engl. *self-organizing feature maps, SOFM*) sind dabei eine spezielle Form von neuronalen Netzen, die die Speicherungs- und Assoziationsfähigkeiten des menschlichen Gehirns nachbilden. So werden z.B. im auditiven Kortex bei der Wahrnehmung von Tonsignalen, Signale mit ähnlichen Tonhöhen durch benachbarte Nervenzellen registriert. Eine derartige Erhaltung räumlicher Zusammenhänge bildet die Grundlage für die Merkmalskarten.
Kohonen [1] beschreibt mit seinem Modell der selbstorganisierenden Merkmalskarten ein Verfahren für die Bildung solcher Merkmalskarten. Es wird dabei eine Folge von Adaptionsschritten durchlaufen, in denen zufällig ein Strom von Eingabesignalen (engl. *Samples*) präsentiert wird und anhand derer sich eine stetige Abbildung von der Gesamtheit der Eingabesignale auf die Neuronenstruktur bildet. Für das hier vorliegende Problem können daher die Daten im Gerber-Format als Eingabedaten des Netzes benutzt und auf eine Neuronenstruktur abgebildet werden. Aufgrund der sich ergebenden Nachbarschaftsbeziehungen der Neuronen im Netz kann dann auch auf räumliche Zusammenhänge zwischen den Eingabedaten geschlossen werden.
Da jedoch das Verfahren von Kohonen von einer statischen Topologie der Neuronen ausgeht, ist der ursprüngliche Algorithmus nur bedingt für diese Problemstellung einsetzbar. Für die Lösung des Problems wurde daher eine Variante des Kohonennetzes verwendet, die sowohl eine dynamische Verbindungsstruktur als auch eine dynamische Anzahl von Neuronen aufweist. Dieser Ansatz soll in den nachfolgenden Abschnitten näher erläutert werden.

## 2.2 Der Algorithmus DCS-GCS

Für die gegebene Problemstellung der Erkennung von zusammenhängenden und getrennten Objekten trotz ungenauer Eingangsdaten schien das neuronale Verfahren DCS-GCS in [2] erfolgversprechend. Dieses Verfahren baut auf dem Prinzip von Kohonen zur Topologierepräsentation und den Ergebnissen von Fritzke [3] über wachsende Zellstrukturen auf. Es arbeitet epochenweise, wobei in jeder Epoche alle Sampledaten dem Netz präsentiert werden, und nach der Epoche das Netz um ein Neuron erweitert wird. Die Verbindungen zwischen den Neuronen werden durch eine N×N Matrix von Verbindungsgewichten beschrieben, wobei mit N die Anzahl der Neuronen im Netz bezeichnet wird. Alle Neuronen sind untereinander mit einem bestimmten Verbindungsgewicht verbunden.
Bei jeder Präsentation eines Sammelpunkts wird die Nachbarschaft des dem Samplepunkt nächstgelegenen Neurons nach einem Kohonen-Lernverfahren angepaßt, wobei Nachbarschaft hier heißt, daß das Verbindungsgewicht einen bestimmten Schwellwert überschreitet. Das Verbindungsgewicht der beiden dem Samplepunkt nächstgelegenen Neuronen wird erhöht und alle anderen Verbindungen in der Matrix abgeschwächt. Nach jeder Epoche wird ein neues Neuron an der Stelle mit der schlechtesten Repräsentation der Topologie durch das Netz eingefügt.
Dieses Verfahren war aus drei Gründen nicht für die benötigte Anwendung geeignet. Alle Kohonen ähnlichen Verfahren haben das Problem, getrennte Objekte auch durch nicht verbundene Neuronen zu repräsentieren. Die Verfahren stellen zwar die Topologie der Sampledaten sehr gut dar, können einzelne getrennte Objekte innerhalb der Topologie aber nicht also solche erkennen. Erst nach sehr vielen Lernschritten werden Verbindungen zwi-

schen Neuronen, die sich auf getrennten Objekten befinden, endgültig gelöscht. Durch die Nachbarschaftsbeziehung der Neuronen werden sie bei den Lernschritten zusammenhängend bewegt, was eine verzerrte Darstellung der Topologie bewirkt.
Ein anderes Problem war der Speicherbedarf, der wegen der Verbindungsmatrix bei $O(M+N+N^2)$ liegt, wobei mit *M* die Anzahl der Sampledaten bezeichnet wird. Dies würde nur die Verwendung von maximal etwa 1000 Neuronen erlauben.
Das größte Problem, das den Einsatz dieses Verfahrens verhinderte, war der immense Rechenaufwand. Die Laufzeit einer Epoche ist $O(M \cdot N^2 + N)$, wobei zusätzlich pro Lernschritt aufwendige mathematische Operationen nötig sind. In Tab. 1 ist als Beispiel die Laufzeit für eine typische Anwendung dieses Algorithmus mit 10.000 Samplepunkten und 1.000 Neuronen dargestellt. Als Vergleich sind die Daten für das neue Verfahren UFDCS aufgetragen.

| Algorithmus | DCS-GCS | UFDCS[1] |
|---|---|---|
| Rechenzeit in Operationen | **~$10^{10}$** | **~$10^5$** |
| Speicherbedarf in Einheiten | **~$10^6$** | **~$10^4$** |

***Tab. 1: Beispiel für 10.000 Samplepunkte und 1.000 Neuronen***

## 2.3 Der Algorithmus UFDCS

Aus den oben genannten Gründen wurde der DCS-Algorithmus an Laufzeit- und speicherkritischen Stellen optimiert bzw. komplett geändert. Im Folgenden wird der verbesserte Algorithmus mit *UFDCS* (Ultra Fast Dynamic Cell Structure) bezeichnet. Hierbei wird ein Neuron durch seinen Gewichtsvektor und das Approximationsgütemaß definiert. Der Gewichtsvektor wird dabei durch die Position (x- und y-Koordinate) des Neurons repräsentiert. Das Approximationsgütemaß beschreibt die Genauigkeit der Darstellung der Sampledaten durch das neuronale Netz, wobei hohe Werte für Stellen mit starker Dynamik im Netz stehen. Bereiche mit großem Approximationsgütemaß deuten auf eine schlechte Identifikation der Sampledaten durch das Netz hin.

### 2.3.1 Kurzbeschreibung des Algorithmus

Nach der Initialisierung des Netzes beginnt die Hauptschleife des Algorithmus. In jedem Durchlauf werden alle Punkte aus den Sampledaten dem neuronalen Netz präsentiert und dabei die Gewichte der Neuronen angepaßt. Eine komplette Präsentation der Sampledaten wird auch als Epoche bezeichnet. Nach jeder Epoche wird das neuronale Netz um ein Neuron erweitert.
Solange das Abbruchkriterium, das je nach Anwendung des Algorithmus adaptiert wird, noch nicht erfüllt ist, wird diese Schleife durchlaufen, wobei das Netz in jeder Epoche um ein Neuron wächst. Durch die Erhöhung der Neuronenanzahl steigt auch die Approximationsgüte des neuronalen Netzes.

[1] Bei Verwendung von Voronoi-Diagrammen.

### 2.3.1.1 Initialisierung

Der Algorithmus beginnt mit der zufälligen Auswahl zweier Punkte aus der Menge der Sampledaten. Auf deren Position werden zwei Neuronen erzeugt, die miteinander verbunden werden.

### 2.3.1.2 Durchlauf einer Epoche

Zunächst wird nun die Prozedur in jeder Epoche betrachtet:
Zuerst werden alle bestehenden Verbindungen zwischen den Neuronen gelöscht, d.h. alle Neuronen sind zu Beginn jeder Epoche isoliert. Die Approximationsgüte jedes Neurons wird auf 0 gesetzt. Damit wird sichergestellt, daß das Netz komplett neu bewertet wird.

Jeder einzelne Punkt m der Sampledaten wird dem Netz präsentiert, wobei die Reihenfolge in der die Sampledaten präsentiert werden, jedesmal zufällig neu gewählt wird. Für jeden Punkt werden zuerst die beiden am nächsten liegenden Neuronen $n_l$ und $n_k$ gesucht und miteinander verbunden (siehe Abb. 3)

$$\|m - n_l\| \cdot \|m - n_k\| \le \|m - n_i\| \cdot \|m - n_j\|, \forall (1 \le i \le N,\ 1 \le j \le N,\ i \ne j),\ l \ne k, \qquad (2.1)$$

$\|m - n_i\|$ ist die Aktivierung des Neurons $n_i$ durch den Sammelpunkt m, hier der euklidische Abstand von m und $n_i$.

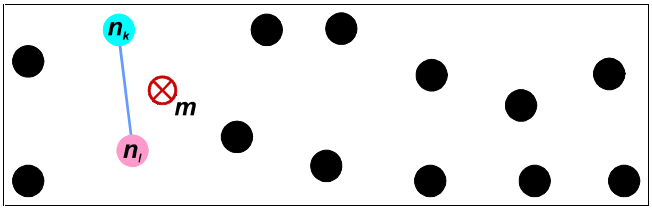


***Abb. 3: Verbindung der nächsten Neuronen***

Der nächste Nachbar des präsentierten Samples, hier mit $n_l$ bezeichnet, wird nun einem beschränkten Kohonenlernschritts unterzogen. Dabei werden die Gewichtsvektoren dieses Neurons und seiner unmittelbaren Nachbarn, d.h. alle mit ihm direkt verbundenen Neuronen, angepaßt (siehe Abb. 4):

$$\Delta n_l = \varepsilon_B \cdot (m - n_l),$$
$$\Delta n_j = \varepsilon_{Nh} \cdot (m - n_i), \forall j \in Nh(n_l), \qquad (2.2)$$

$Nh(n_l)$ bezeichnet hierbei die direkt verbundenen Nachbarn von Neuron $n_l$. Im Prinzip werden die Neuronen jeweils um den Faktor $\varepsilon_B$ oder $\varepsilon_{Nh}$ auf die Position des Sammelpunkts hinbewegt.

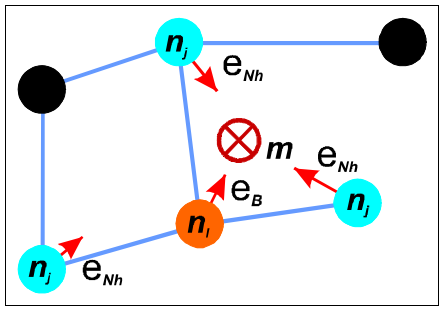


***Abb. 4: Anpassung der Gewichtsvektoren der Neuronen***

Danach wir das Approximationsgütemaß des nächsten Nachbarn angepaßt, d.h. um das Quadrat des Abstands zum Samplepunkt erhöht:

$$R(n_l) = R(n_l) + \|n_l - m\|^2, \tag{2.3}$$

wobei $R(n_l)$ das Approximationsgütemaß des Neurons $n_l$ bezeichnet.

Dieser Algorithmus wird nun für jeden einzelnen Samplepunkt durchgeführt. Die Verbindungen der Neuronen werden dabei nach und nach wieder aufgebaut. Da die Anzahl der Sampledaten im Regelfall wesentlich größer als die Anzahl der Neuronen ist, wird die Verbindungsstruktur beim Präsentieren rasch wieder aufgebaut, so daß die meisten Sampledaten einem schon verbundenen Netz präsentiert werden.
Der Grund für dieses Vorgehen liegt darin, daß bei der gegebenen Problemstellung gleichermaßen zusammenhängende und getrennte Bereiche zu erkennen sind. Fehlerhafte Verbindungen, die an sich getrennte Bereiche verbinden und so die Anpassung des Netzes an die richtige Struktur der Sampledaten hemmen, werden von vornherein nicht aufgebaut, wenn die Approximationsgüte groß genug ist, also genügend Neuronen den Bereich darstellen. Bei anderen Ansätzen müßten die Gewichte der falschen Verbindungen durch das Lernen des Netzes abgeschwächt werden. Dies aber würde die Lerndauer wesentlich erhöhen. Nötige Verbindungen, die zusammenhängende Bereiche darstellen, werden schon nach der Präsentation des ersten Sammelpunkts aufgebaut, der diesen Zusammenhang beschreibt.

#### 2.3.1.3 Einfügen eines neuen Neurons

Nach dem Ende einer Epoche, also der Präsentation aller Sampledaten, wird ein neues Neuron erzeugt und in das Netz eingefügt:

1. Dazu wird zuerst das Neuron $n_r$ mit dem insgesamt höchsten Wert des Approximationsgütemaßes gesucht. An der Position dieses Neurons ist nach der Definition des Approximationsgütemaßes die Repräsentation der Sampledaten durch die Neuronen des Netzes am ungenauesten. Unter seinen direkt verbundenen Nachbarn wird das Neuron mit dem lokal höchsten Wert gesucht und als $n_t$ bezeichnet (siehe Abb. 5):

$$R(n_r) \geq R(n_j),\ \forall (1 \leq j \leq N)$$
$$R(n_t) \geq R(n_i),\ \forall i \in Nh(n_r) \tag{2.4}$$

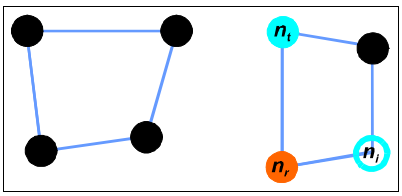


***Abb. 5: Auswahl des Neurons mit höchstem Approximationsgütemaßes***

2. Die Position des einzufügenden Neurons $n_u$ wird gemäß dem Verhältnis der Approximationsgütemaße der beiden Neuronen bestimmt (siehe Abb. 6):

$$\tau = \frac{R(n_t)}{R(n_r) + R(n_t)},$$
$$n_u = n_r + (n_t - n_r) \cdot \tau \tag{2.5}$$

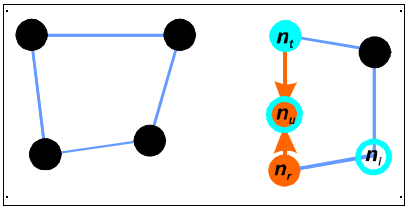


***Abb. 6: Einfügen des neuen Neurons***

Dadurch wird sichergestellt, daß das neu erzeugte Neuron n in dem Bereich des Testfeldes liegt, an welchem die Repräsentation der Sampledaten durch die Neuronen des Netzes am ungenauesten ist. Das Approximationsgütemaß beschreibt die Summe der Abstandsquadrate der Samplepunkte $m_i$ von n:

$$R(n) = \sum_{\substack{i=\text{alle nächsten} \\ \text{Samplepunkte}}} \|n - m_i\|^2 \qquad (2.6)$$

Die Verbindung zwischen den beiden Neuronen $n_t$ und $n_r$ wird getrennt und dafür jeweils eine Verbindung zu dem neu eingefügtem Neuron $n_u$ aufgebaut. Da aber Neuronen in der verwendeten Implementierung des Algorithmus nur zwischen zwei Epochen eingefügt werden und alle Verbindungen vor jeder Epoche gelöscht werden, hat das hier keine Auswirkung.

### 2.3.2 Wahl des Abbruchkriteriums

Beim Durchlauf des Algorithmus werden fortwährend neue Neuronen in das neuronale Netz eingefügt und das Netz insgesamt der Struktur der Sampledaten angepaßt. Die Darstellung der Topologie durch das Netz wird dadurch sukzessive genauer. Wenn eine für die Anwendung ausreichende Genauigkeit der Repräsentation erreicht ist, wird der Algorithmus beendet, und das neuronale Netz hat die Struktur gelernt.
In der hier verwendeten Anwendung besteht die Hauptaufgabe des neuronalen Netzes darin, zu identifizieren, ob zwei räumlich getrennte Bereiche auf der Leiterplatte wirklich zwei getrennte Objekte, wie z.B. Leiterbahnzüge, darstellen, oder zu einem Objekt gehören und nur durch Mängel in der Beschreibung isoliert sind. Solche Ungenauigkeiten treten z.B. durch Rundungsfehler auf, wenn eine Leiterbahn vor ihrem Lötpunkt endet und ihn nicht berührt.
Leiterbahnen haben aus physikalischen Gründen eine bestimmte Mindestdicke und Mindestabstand voneinander. Dies wird ausgenutzt, um das Abbruchkriterium zu definieren. Das neuronale Netz stellt die Topologie der Leiterplatte genügend genau dar, wenn die Entfernung von jedem Neuron zu seinen direkt verbundenen Nachbarn kleiner ist als der physikalisch bedingte Mindestabstand einzelner Objekte. Die maximale Entfernung der Neuronen wird beim Verbinden der Neuronen aktualisiert, so daß nach jeder Epoche nur geprüft werden muß, ob dieser Wert kleiner als der vorher definiert Schwellwert D ist.

$$\|n_j - n_i\| \le \tfrac{1}{2} D,\ \forall (1 \le j \le N), \forall i \in Nh(n_r) \qquad (2.7)$$

wobei der Schwellwert D den Mindestabstand zweier Objekte darstellt und $n_r$ das Neuron mit dem höchsten Wert des Approximationsgütemaßes ist.

## 2.4 Benchmarking

Da das DCS-GCS Verfahren wegen seiner langen Rechenzeit und dem hohen Speicherbedarf nicht geeignet für die beschriebene Problemstellung war, mußte ein besseres Verfahren entwickelt werden. Bei der Anwendung beträgt die Anzahl der Samplepunkte viele tausend

und die der Neuronen bis zu einigen tausend. Dies erfordert eine Reduktion des Speicherbedarfs und der Rechenzeit um Größenordnungen.
Im folgenden wird die Rechenzeit des Algorithmus untersucht.

### 2.4.1 Rechenzeit

Da in allen Epochen jeder Samplepunkt dem Netz präsentiert werden soll, ergibt sich für die minimale Rechenzeit $O(M \cdot X + Y)$, wobei X den Aufwand für das Lernen des Netzes bei jeder Präsentation eines Punktes und Y den Aufwand für das Einfügen eines Neurons und das Prüfen des Abbruchkriteriums nach jeder Epoche darstellt. Besonders der Rechenaufwand für das Lernen des Netzes muß minimiert werden, da er multipliziert mit der Anzahl der Samplepunkte in die Laufzeit eingeht. Bei der Präsentation eines Sammelpunkts benötigt nur die Suche nach den nächsten Nachbarn eine nicht konstante Rechenzeit. Ein naiver Ansatz zur Suche des nächsten Nachbarn ergibt im planaren Fall einen Aufwand von $O(N)$. Diese Suche läßt sich aber durch Verwendung von Voronoi-Diagrammen wesentlich effizienter gestalten. Bei vorliegendem Voronoi-Diagramm lassen sich die nächsten Nachbarn zu einem beliebigen Punkt in $O(\log(N))$ suchen. Zum Aufbau eines Voronoi-Diagramms ist ein Rechenaufwand von $O(N \cdot \log(N))$ nötig. Da aber das neuronale Netz sukzessive aufgebaut wird, kann das Voronoi-Diagramm durch Einfügen des jeweils neuen Neurons in das bestehende Diagramm erzeugt werden. Dabei fällt nur konstante Rechenzeit an (siehe [4]). Nur beim Start des Algorithmus muß ein Voronoi-Diagramm komplett erzeugt werden, was aber bei den zwei Neuronen der Initialisierung nur einen Aufwand von $O(1)$ ergibt. Der Aufwand für eine Epoche reduziert sich bei Verwendung von Voronoi-Diagrammen also insgesamt von $O(M \cdot N)$ auf $O(M \cdot \log(N))$.
Die Initialisierung des Approximationsgütemaßes vor jeder Epoche hat einen Rechenbedarf von $O(N)$, da alle Neuronen bearbeitet werden müssen.
Beim Einfügen eines neuen Neurons in das Netz nach jeder Epoche muß das Neuron mit dem höchsten Wert des Approximationsgütemaßes gesucht werden. Dies erfordert einen Aufwand von $O(N)$. Die Einfügeoperation selbst und das Erweitern des Voronoi-Diagramms um ein Neuron benötigen konstante Zeit.
Das Prüfen des Abbruchkriteriums benötigt nur konstante Zeit, da die maximale Distanz zwischen zwei verbundenen Neuronen beim Aufbau dieser Verbindungen aktualisiert wird.
Insgesamt entsteht für einen Durchlauf einer Epoche bei Verwendung von Voronoi-Diagrammen ein Rechenaufwand von $O(M \cdot \log(N))$ zuzüglich dem Aufwand für die oben beschriebene Bearbeitung von $O(N)$. Insgesamt ist die Laufzeit also $O(M \cdot \log(N) + N)$.

### 2.4.2 Speicherbedarf

Ebenso wichtig wie die Rechenzeit ist der Speicherbedarf des Algorithmus. Der unvermeidliche Platzbedarf für die Sampledaten und die Neuronen ist $O(M + N)$ und kann nicht unterschritten werden. Zusätzlicher Speicher wird bei neuronalen Netzen vor allem für die Darstellung der Verbindungen zwischen den Neuronen benötigt. Da in diesem Algorithmus nur Verbindungen zu direkten Nachbarn verwendet werden, wird keine Verbindungsmatrix benötigt. Durch die Verwendung von rein binären Verbindungen spart man auch die Verbindungsgewichte ein und muß lediglich die Menge der direkt verbunden Nachbarn zu jedem Neuron speichern. Die Anzahl der direkten Nachbarn ist bei dem verwendeten planaren System in der Größenordnung zwei bis fünf, auf jeden Fall aber unabhängig von N. Der Gesamtspeicherbedarf des Algorithmus ist aus diesem Grund $O(M + N)$.

### 2.4.3 Vergleich mit DCS-GCS

In Tab. 2 ist ein Vergleich von UFDCS mit dem Verfahren DCS-GCS dargestellt. Wie man erkennen kann, ist sowohl die Rechenzeit, als auch der Speicherbedarf um Größenordnungen bei UFDCS besser. In Tab. 1 wird die Rechenzeit und der Speicherbedarf für eine typische Anwendung mit 10.000 Samplepunkten und 1000 Neuronen gezeigt.

| Algorithmus | DCS-GCS | UFDCS[2] |
|---|---|---|
| Rechenzeit | $O(M \cdot N^2 + N)$ | $O(M \cdot \log(N) + N)$ |
| Speicherbedarf | $O(M + N + N^2)$ | $O(M + N)$ |

***Tab. 2: Vergleich von DCS-GCS mit UFDCS***

# 3 Anwendung von UFDCS auf Gerber-Daten

## 3.1 Einlesen und Initialisierung

Für eine Anwendung des UFDCS-Algorithmus auf reale Platinendaten muß zunächst eine geeignete Verteilung der Testpunkte gefunden werden. Die Daten der meisten PCBs liegen im sog. *Gerber-Format* vor [5] Dabei handelt es sich, wie bereits in Abschnitt 1.1 beschrieben, um ein relativ simples Plotterdatenformat mit nur wenigen Steuerdaten. Eine Vielzahl unterschiedlicher Dialekte und Abweichungen vom Standard erschwert jedoch die korrekte Interpretation.
Zusammen mit den Blendendaten für die Plottersteuerung werden aus den Gerber-Daten zweidimensionale geometrische Objekte generiert (im Folgenden *Gerber-Objekte* oder kurz *GOBs* genannt), auf denen dann Testpunkte für die Eingabe an den UFDCS-Algorithmus erzeugt werden.
Die Hauptaufgabe des Einleseschritts ist die Identifizierung geometrischer Objekte durch Interpretation der gegebenen Gerber-Daten. Im wesentlichen besteht die Menge der Objekte aus Quadraten bzw. Rechtecken (SMDs), Kreisen (Lötpunkte und Bohrungen) und Linienzügen (leitende Verbindungen auf der Platinenoberfläche).
Einfach zu interpretieren sind hierbei Rechtecke und Kreise, die technisch durch Belichten mittels Schablonen auf die eigentliche Platine übertragen werden. Die Abmessungen dieser Schablonen stehen im Kopfteil der zu interpretierenden Gerber-Datei und sind daher direkt auf die Abmessungen der geometrischen Objekte übertragbar.
Schwieriger ist die Interpretation von gezeichneten Objekten (oft als sog. *Drawn Pads* bezeichnet). Diese setzen sich aus vielen, eng beieinanderliegenden Linienzügen zusammen und können eine Einhüllende von beliebiger Form besitzen. Erschwerend kommt hinzu, daß sich solche zusammengehörenden Linienzüge nicht zwingend berühren müssen. Eine Lösung dieses Problems wird im Folgenden näher betrachtet.

## 3.2 Verteilung der Testpunkte

Die Verteilung der Testpunkte auf den identifizierten geometrischen Objekten ist entscheidend für die Korrektheit und Laufzeit des UFDCS-Algorithmus. Da die primäre Aufgabe darin besteht, zweidimensionale geometrische Objekte zu erkennen, liegt eine zufällige,

[2] Bei Verwendung von Voronoi-Diagrammen.

zweidimensionale Verteilung nahe. Ebenso kann dadurch die Anwendbarkeit des ursprünglichen Algorithmus von Bruske und Sommer [2] auf reale Objektdaten getestet werden. Wesentliche Laufzeitverbesserungen wurden durch geänderte Testpunktverteilungen, die nur noch Teilbereiche der Objekte berücksichtigen, erreicht.

### 3.2.1 Zufällige zweidimensionale Verteilung

Bei der zufälligen zweidimensionalen Punktverteilung werden Testpunkte auf die Oberfläche der GOBs verteilt. Die Dichte der Punkte wird dabei durch die Intensität des punkterzeugenden Prozesses gesteuert und ist ein Parameter des gesamten Verfahrens. Die Intensität muß dabei so gewählt werden, daß auch feinste Strukturen der Leiterplatte noch aufgelöst werden können.
Zunächst wird keine Clusterung der Punkte auf der Oberfläche der GOBs berücksichtigt, d.h. alle Testpunkte werden gleichmäßig verteilt. Daß eine Clusterung durchaus sinnvoll sein kann, wird in Abschnitt 3.2.3 gezeigt. Ist das Abbruchkriterium erfüllt, so endet der Algorithmus mit dem Ergebnis, daß alle zusammenhängenden Objekte mit einem Netz aus Neuronen und Verbindungen dazwischen überdeckt sind; genauer: das Innere aller ausgebildeten Verbindungen eines Neuronennetzes approximiert die Fläche der geometrischen Objekte.
Durch die Verteilung der Testpunkte auf der gesamten Fläche der GOBs ist die Laufzeit des Verfahrens proportional zur Flächendichte der Objekte auf der Platinenoberfläche. Bei gleicher Oberflächendichte wächst also die Anzahl der generierten Testpunkte quadratisch mit den Abmessungen der Platine. Dadurch ist für größere Platinen diese Vorgehensweise nahezu ungeeignet. Bei Platinen, die großflächige leitende Verbindungen und eine hohe Oberflächendichte aufweisen nimmt die Anwendbarkeit des Algorithmus weiterhin ab. Oftmals existieren auf Platinen großflächige Massenleitungen (ground nets, GND), die aufgrund elektrischer Leitfähigkeit den von anderen Objekten ungenutzten Platz auf der Platine einnehmen. Die Anzahl der verwendeten Testpunkte steigt dadurch enorm.

### 3.2.2 Randpunktverteilung

Wenn man davon ausgehen kann, daß die Ausgangsdaten fehlerfrei vorliegen und nicht etwa durch Rundungsfehler o.ä. verfälscht wurden, so könnten maximale Zusammenhangskomponenten durch Schnitt der Objekte berechnet werden. Da die einzelnen Objekte selbst zusammenhängend sind, genügt es, den Schnitt der Ränder der Objekte zu betrachten. Der Algorithmus wäre damit ein Spezialfall des allgemeinen Polygonschnittproblems. Geht man davon aus, daß eine konstante obere Schranke für die Anzahl der Linienzüge eines Objekts existiert, so liegt die Worst-Case Beschreibungskomplexität des Ergebnisses in $O(n^2)$, wobei n die Anzahl der GOBs ist. Bei hinreichend „gutmütigen“ Objekten lassen sich jedoch schnellere Algorithmen finden, etwa Line-Sweep-Verfahren [4].
Im Gegensatz zu Abschnitt 3.2.1 werden nun Testpunkte nur auf dem Rand der zu betrachtenden Objekte verteilt. Da die fraktale Dimension der betrachteten Objekte immer gleich 2 ist, wächst die Anzahl der Testpunkte dadurch nur noch linear mit der Anzahl der Objekte. Dies könnte auch durch eine obere Schranke für die maximal zulässige Flächendichte der Samplepunkte erreicht werden. Zu unregelmäßige oder fraktale Objekte würden somit durch die Punktverteilung geglättet werden.
Durch den Algorithmus wird mit dieser Vorgehensweise die Einhüllende der einzelnen Objekte berechnet. Berühren bzw. schneiden sich zwei Objekte, so gehen die an den Schnittpunkten gebildeten Neuronen Verbindungen zu beiden Objekten ein. Abhängig von der Laufzeit des Algorithmus (also im Wesentlichen durch ein geeignetes Abbruchkriterium,

siehe Abschnitt 2.3.2) werden auch Verbindungen zu ausreichend nahen, jedoch getrennten Objekten, wie sie durch Fehler in der Beschreibungssprache bzw. Rundung vorkommen können, aufrecht erhalten. Begünstigt wird dieser Effekt durch die erhöhte Flächendichte der Testpunkte in diesen Bereichen, da hier die Wahrscheinlichkeit für die Auswahl eines Testpunkts durch den Algorithmus höher ist als in Bereichen mit entfernt zueinander liegenden GOBs.

### 3.2.3 Hot-Spot-Verteilung

Bei den oben vorgestellten Verfahrensweisen entstehen sehr viele Neuronenverbindungen nur zwischen Neuronen des gleichen GOBs. Für die effiziente Berechnung der maximalen Zusammenhangskomponenten sollten sich idealerweise nur Verbindungen zwischen unterschiedlichen GOBs bilden.
Bei zusätzlichem Vorwissen über die verwendeten Gerber-Daten läßt sich ein weiterer wesentlicher Geschwindigkeitsgewinn erzielen, falls nur Bereiche mit Samplepunkten versorgt werden, die eine erhöhte Wahrscheinlichkeit besitzen, Verbindungen zwischen GOBs auszubilden. Vor allem bei Linienzügen werden hier sehr viele Testpunkte „verschwendet".
Zunächst werden alle Objekte mit relativ geringen Ausdehnungen komplett mit Testpunkten versorgt, d.h. SMDs, Bohr- und Lötpunkte werden mit Testpunkten gemäß der Randpunktverteilung in Abschnitt 3.2.2 bestückt. Dann werden die Enden der Linienzüge betrachtet und bis zu einer Länge $c \cdot b$ mit Randtestpunkten versorgt. Dabei ist b die Breite des zu versorgenden Linienzugs und c eine Konstante des Verfahrens. Auf diese Weise werden alle Leiterbahnenden mit Testpunkten besetzt, falls ihre Länge $2 \cdot (c \cdot b)$ überschreitet. Leiterbahnen, die kürzer als $2 \cdot (c \cdot b)$ sind, werden wie in Abschnitt 3.2.2 komplett versorgt. Damit wird vermieden, daß sich um die Mitte von Leiterbahnzügen Testpunkte häufen, sich dadurch der Algorithmus wiederum verlangsamt und durch die Überrepräsentation in „unwichtigen" Bereichen andere Berührungs- und Schnittpunkte vernachlässigt werden.
Typischerweise werden durch diese Vorgehensweise alle kritischen Regionen, d.h. Bereiche, in denen Schnittpunkte auftreten können, versorgt. Kreuzungspunkte zweier Leiterbahnen, die nicht in den Bereich der Enden fallen, treten praktisch nicht auf. Jedoch kann dies nicht völlig ausgeschlossen werden. Deshalb werden mögliche Regionen durch einen Vorverarbeitungsschritt identifiziert und die entsprechenden Bereiche mit Samplepunkten belegt. Nach Beendigung des Algorithmus repräsentieren die Neuronenverbindungen wiederum die zusammenhängenden Leitungsnetze.

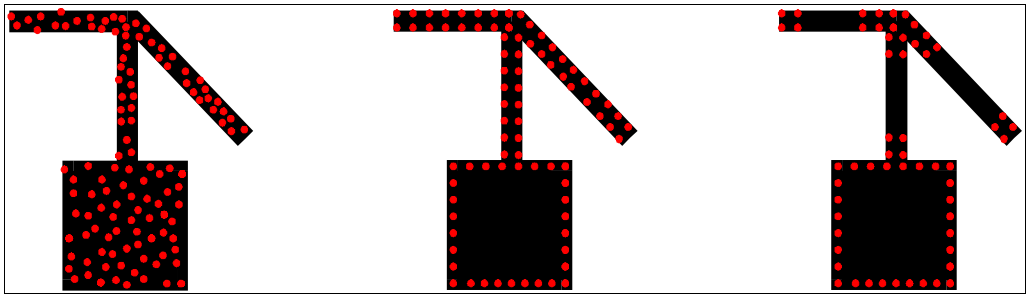

***Abb. 7: Schematische Darstellung der Punktverteilungen***

## 3.3 Kachelung der Leiterplatte

In anfänglichen Untersuchungen zeigte sich, daß zur korrekten Interpretation einer typischen Leiterplatte die Anzahl der Samplepunkte für die Berechnung in einem Schritt zu groß war. Selbst bei der Verwendung der Hot-Spot-Verteilung aus Abschnitt 3.2.3 konnte die gesamte Platine nicht in abwartbarer Zeit interpretiert werden. Die untersuchte Leiterplatte wies etwa 50.000 einzeln identifizierbare Objekte auf (ein Ausschnitt aus der gekachelten Leiterplatte zeigt Abb. 8. Um die Anzahl der Testpunkte zu verringern, mußte die Platine in unterschiedliche Bereiche aufgeteilt werden.
Um möglichst wenig Vorwissen über die Struktur der Objekte vorauszusetzen, wurde die Einteilung erst nach der Versorgung mit Testpunkten vorgenommen. Es wäre auch denkbar, zunächst die geometrischen Objekte einem Bereich auf der Platine zuzuschlagen und dann die Verteilung der Samplepunkte vorzunehmen. Gegen diese Vorgehensweise sprach vor allem die Tatsache, daß mit zunehmend feinerer Kachelung der Leiterplatte (im Folgenden meint Kachelung immer die Überdeckung der Leiterplatte mit zweidimensionalen Teilmengen) häufiger Objekte auftreten, die mehrere Kacheln schneiden. Die Entscheidung darüber, welcher Kachel diese Objekte zugeschlagen werden müssen, ist entscheidend für die Korrektheit des Verfahrens. Um eine korrekte Interpretation zu ermöglichen, muß ein solches Objekt logischerweise zu beiden Kacheln gerechnet werden. Durch sukzessive Aufteilung kann sich dadurch die Anzahl der insgesamt tatsächlich zu betrachtenden Objekte drastisch erhöhen, bzw. die Anzahl der Objekte pro Kachel nicht weiter verringern lassen.
Ebenso muß ein geeignetes Maß dafür gefunden werden, wann die Kachelung einer Platine ausreichend fein ist. Wird die Kachelung erst nach der Verteilung der Testpunkte vorgenommen, beschreibt die Anzahl der Testpunkte, die von einer Kachel überdeckt werden, den zu erwartenden Berechnungsaufwand sehr genau. Wird die Zuordnung der Objekte zu Kacheln vor der Verteilung der Samplepunkte berechnet, so muß sukzessive eine Bewertung des Berechnungsaufwandes aufgrund der Zuteilung der Objekte berechnet werden. Erschwert wird dies wiederum dadurch, daß Objekte Kachelungsgrenzen überschreiten können und somit nur teilweise in die Bewertung einfließen dürfen. Ein Maß wäre z.B. der Flächen- oder Umfangsanteil, der durch die Hinzunahme eines Objekts zu einer Kachel zusätzlich entstünde. Geht man aber von einer Hot-Spot-Verteilung aus, so müßte zusätzlich berechnet werden, wieviel von diesem Anteil wieder in ein Gebiet fällt, das von der Hot-Spot-Verteilung berücksichtigt wird.
Da die betrachteten Mengen jedoch vorsortiert sind, genügt es, pro Rekursionsschritt lediglich für jede der sortierten Mengen zwei Indizes zu speichern, welche die Grenzen der für diese Kachel zu betrachtenden Punktmenge angeben. Die ursprüngliche Menge der Testpunkte muß also nicht kopiert werden und die tatsächlichen Testpunkte der Kachel ergeben sich als Schnitt der Punkte beider Indexbereiche.

| | |
|---|---|
| **Algorithmus:** | Tile |
| **Beschreibung:** | Kachelt die Menge der Samplepunkte. |
| **Parameter:** | samples_x, samples_y: Nach Koordinaten sortierte Mengen der Samplepunkte.<br>left_x, right_x: Indizes der nach x-Koordinate sortierten Samplepunkte.<br>lower_y, upper_y: Indizes der nach y-Koordinate sortierten Samplepunkte.<br>which: Gibt an, welche Koordinate geteilt werden soll.<br>Θ: Schwellwert für die Anzahl der Samplepunkte |
| **Startwerte:** | left_x, right_x, lower_y, upper_y beschreiben jeweils die gesamten sortierten Samplemengen. |
| **Ausgabe:** | Kachel, deren Anzahl von Testpunkten erstmals kleiner als Θ ist. |

```
Tile (samples_x, samples_y, left_x, right_x, lower_y, upper_y, which)
{
        if (points_in_section (samples_x, samples_y, left_x, right_x, lower_y, upper_y) > Θ)
        {
                if (which == x)
                {
                        cut_x = find_good_cutting (samples_x, left_x, right_x);

                        Tile (samples_x, samples_y, left_x, cut_x, lower_y, upper_y, y);
                        Tile (samples_x, samples_y, cut_x, right_x, lower_y, upper_y, y);
                }
                else
                {
                        cut_y = find_good_cutting (samples_y, lower_y, upper_y);

                        Tile (samples_x, samples_y, left_x, right_x, lower_y, cut_y, x);
                        Tile (samples_x, samples_y, left_x, right_x, cut_y, upper_y, x);
                }
        }
        else
        {
                write_out (samples_x, samples_y, left_x, right_x, lower_y, upper_y);
        }
}
```

***Algorithmus 1: Rekursiver Kachelalgorithmus***

Zur Aufteilung der Testpunkte in Kacheln wurde ein rekursiver Algorithmus gewählt. Zur Beschleunigung des Verfahrens wird zunächst die Menge der Testpunkte sowohl nach der x-Koordinate als auch nach der y-Koordinate sortiert. Dies kann auch schon beim Erzeugen und Einfügen der Testpunkte in die Menge aller Testpunkte geschehen.
Zu Beginn des Algorithmus liegen alle Testpunkte auf einer einzigen Kachel. Sind zu viele Punkte in dieser Menge, wird abwechselnd nach x-Koordinate und y-Koordinate geteilt und die dadurch entstehenden Mengen weiter rekursiv betrachtet. Wichtig dabei ist, daß nicht bei jedem Aufteilungsschritt neue Mengen gebildet werden. Dies würde zu exponentiell mit der Zahl der Aufteilungsschritte k wachsenden Punktmengen führen. Bezeichnet n die ursprüngliche Anzahl der insgesamt verteilten Testpunkte, so würde die Anzahl der Testpunkte aller Kacheln zusammen nach k Aufteilungen $n \cdot 2^k$ betragen.

### 3.3.1 Geometrische Teilung

Noch nicht näher betrachtet wurde im obigen Algorithmus die Funktion *find_good_cutting*, die das Aufteilen der aktuellen Samplepunkte vornimmt. Idealerweise sollte eine Aufteilung Mengen mit gleicher Mächtigkeit erzeugen und somit die Menge der im nächsten Schritt zu betrachtenden Testpunkte halbieren.
Ein schnell zu implementierendes Verfahren ist die Berechnung der Kachelungsgrenzen als geometrische Mitte der ursprünglichen Kachel. Sind alle Samplepunkte annähernd gleichverteilt, führt dieses Verfahren zu kleinsten Kacheln mit wenig Schwankungen bezüglich der Anzahl der Testpunkte. Jedoch können bei stark geclusterten Testpunktbereichen deutliche Unterschiede in der Testpunktanzahl auftreten. Vor allem bei der Hot-Spot-Verteilung hat sich gezeigt, daß häufig Kacheln mit sehr unterschiedlicher Anzahl an Samplepunkten erzeugt werden.

### 3.3.2 Teilung über Mediansuche

Die oben beschriebenen Nachteile einer geometrischen Aufteilung lassen sich vermeiden, wenn statt dessen eine Medianteilung implementiert wird. Dazu wird bei der Aufteilung nach einer Koordinate das mittlere Element als Teilungsgrenze bestimmt. Da es sich bei den betrachteten Punktmengen um vorsortierte Mengen handelt, ist dies in linearer Zeit möglich. Selbst bei nicht vorsortierter Menge kann eine Mediansuche in linearer Zeit erfolgen [4].
Durch diese Vorgehensweise wird sichergestellt, daß sich die Anzahl der Testpunkte zweier in einem Teilungsschritt entstehender Kacheln höchstens um eins unterscheidet. Für eine Parallelisierung des Verfahrens wird somit sichergestellt, daß annähernd gleich viel Rechenzeit pro Kachel aufgewendet werden muß. Auch kann dadurch nach Berechnung einer Kachel die verbleibende restliche Rechenzeit recht genau bestimmt werden. Warum es dennoch zu kleineren Schwankungen kommen kann, zeigt der nächste Abschnitt.

### 3.3.3 Überlappende Bereiche bei der Kachelung

Da nicht ausgeschlossen werden kann, daß bei der Aufteilung in Kacheln Testpunkte eines GOBs in unterschiedlichen Kacheln zu liegen kommen, muß eine Betrachtung der Randbereiche vorgenommen werden. Verdeutlicht wird dies in Abschnitt 3.4, wo die Interpretation der Neuronen und Verbindungen nach Beendigung des UFDCS-Algorithmus beschrieben wird.
Bei der Aufteilung der Testpunkte wird um jede Kachel herum ein schmaler Randbereich zusätzlich hinzugenommen, d.h. die in diesen Bereich fallenden Punkte werden den eigentlichen Testpunkten der Kachel hinzugefügt. Realisiert wird dies durch eine nachträgliche Indexverschiebung der ursprünglichen Kachelungsgrenzen.
Um nicht in jedem Aufteilungsschritt diese Randpunkte zusätzlich betrachten zu müssen, und insbesondere dadurch die insgesamt als Randbereich entstehende Fläche unnötig zu vergrößern, genügt es, beim letzten Teilungsschritt, also bevor die Testpunkte einer Kachel endgültig festgelegt werden, diese Punkte zu berücksichtigen.
Durch die Berücksichtigung von Rändern können natürlich Schwankungen in der Testpunktanzahl zwischen Kacheln entstehen, wenn Randbereiche mit unterschiedlicher Dichte die Kacheln mehr oder weniger vergrößern.

## 3.4 Interpretationsschritt der Kacheln

Der Interpretationsschritt der Kacheln hat die Aufgabe, aus den Neuronen- und Verbindungsdaten, zusammen mit den ursprünglichen Gerberobjekten, die zu Beginn von Kapitel 3

beschriebenen Leitungsnetzdaten zu erzeugen, um die geforderten Tests (Leitungs- und Isolationstest) durchführen zu können.
Zunächst werden die gegebenen Gerber-Daten wie oben beschrieben eingelesen, Samplepunkte darauf verteilt und die Menge der Samplepunkte in Kacheln aufgeteilt. Auf die Testpunkte jeder Kachel wird nun der UFDCS-Algorithmus angewendet, um die Verbindungsstruktur dieser Kachel zu berechnen. Nachdem das Abbruchkriterium erfüllt ist, repräsentieren die erzeugten Neuronen zusammen mit den Verbindungen die Struktur der Objekte auf der Leiterplatte.

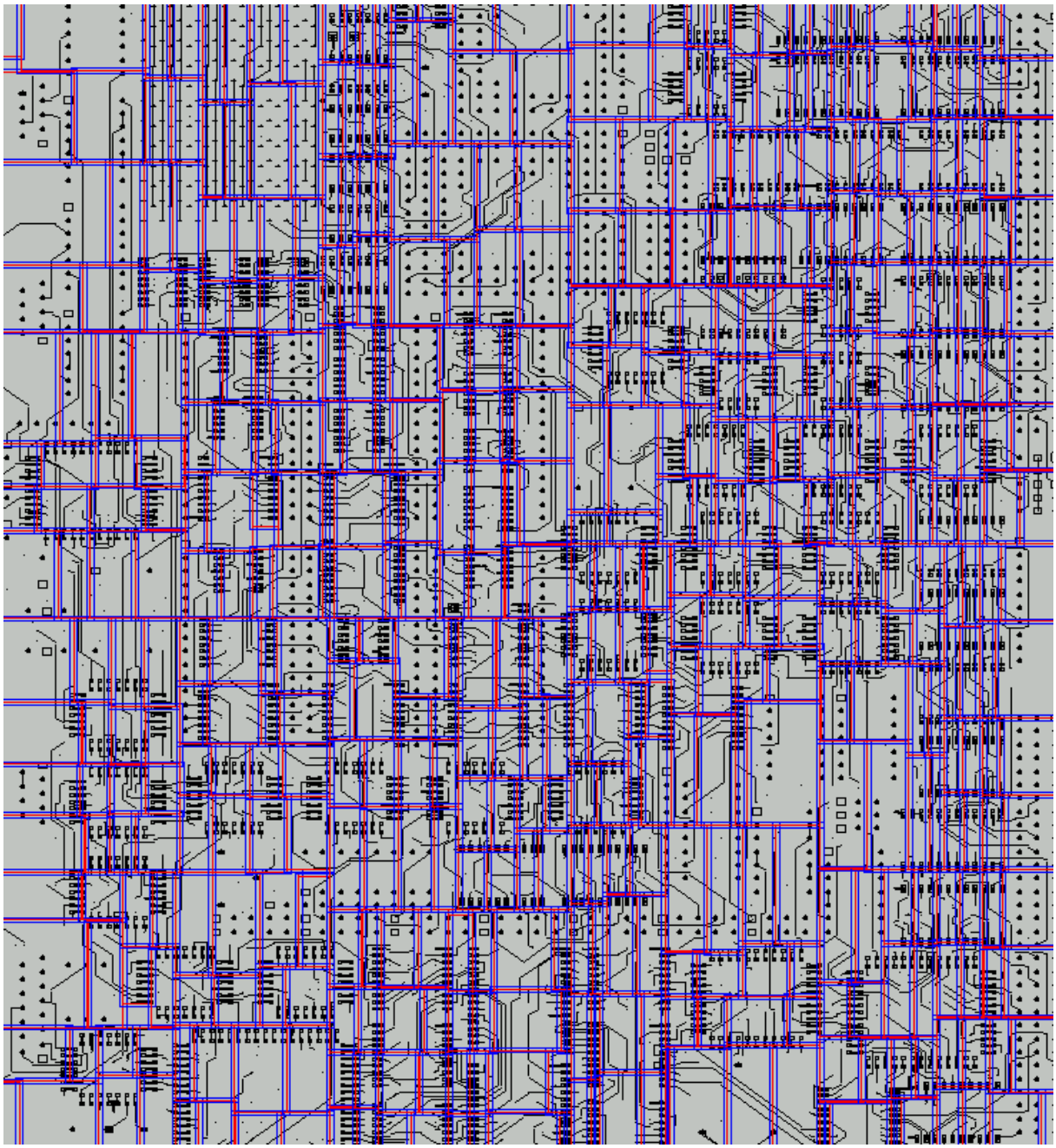

***Abb. 8: Gekachelte Leiterplatte (Medianteilung)***

**Algorithmus:** UFDCS_Gerber
**Beschreibung:** Gesamtalgorithmus zur Interpretation der Gerber-Daten
**Ausgabe:** Maximal zusammenhängende Mengen, die die Leitungsnetze repräsentieren.

```
UFDCS_Gerber() {
        samples = {};
        gobs = read_all_objects ();
        forall (gob g in gobs) {
                points = g.generate_samplepoints ();
                samples.add (points);
        }
        samples_x = samples.sort_x ();
        samples_y = samples.sort_y ();
        nx = samples_x.count ();
        ny = samples_y.count ();
        Tile (samples_x, samples_y, 0, nx, 0, ny, x);
        forall (tile) {
                UFDCS (tile);
                forall (neuron n in neurons) {
                        A_n = n.locate_gobs ();
                        forall (neighbours m of n) {
                                A_n = A_n ∪ m.locate_gobs ();
                        }
                }
                forall (i, j) {
                        if (A_i ∩ A_j ≠ ∅) {
                                A_i.add (A_j);
                                A_j.delete ();
                        }
                }
                forall (A_i ≠ ∅) {
                        write_out (A_i);
                }
        }
        read_all (A_i);
        forall (i, j) {
                if (A_i ∩ A_j ≠ ∅) {
                        A_i.add (A_j);
                        A_j.delete ();
                }
        }
        forall (A_i ≠ ∅) {
                write_out_nets (A_i);
        }
}
```

***Algorithmus 2: UFDCS-Gerber Gesamtalgorithmus***

Um zusammenhängende Objekte zu identifizieren wird für jedes erzeugte Neuron berechnet, ob der damit verbundene geometrische Ort (Punkt) einen Schnitt mit den ursprünglichen GOBs hat und diese in Mengen gespeichert. Nach diesem *Point-Location-Test* werden Mengen mit nichtleerem Schnitt zu größeren Mengen vereinigt. Treten keine weiteren

Schnitte von Mengen auf, sind die maximalen Zusammenhangskomponenten, d.h. die in dieser Kachel enthaltenen Leiterbahnnetze, komplett berechnet.
Alle so gebildeten maximalen Mengen werden zuletzt wiederum nach obigem Verfahren vereinigt. Durch die überlappenden Regionen werden nun auch Leitungsnetze, die über Kacheln hinweg verlaufen, korrekt erkannt. Am Ende des Gesamtalgorithmus entsprechen die maximalen Zusammenhangsmengen den Leitungsnetzen der gesamten Platine.
Ein Nachbearbeitungsschritt berechnet nun noch aus der Kopfinformation der Gerber-Datei zusammen mit den Elementen der maximalen Zusammenhangsmengen die AFT-Repräsentation der Daten und gibt diese in eine Datei aus.
Durch die Verwendung von *Union-Find-Strukturen* [4] kann der Aufwand für die oben beschriebenen Mengenoperationen deutlich gegenüber einer naiven Vorgehensweise verringert werden. Für Mengen A und B mit n bzw. m Elementen kann dadurch der Schnitt beider Mengen in $O(n \cdot \log m)$ berechnet werden, wobei n das Minimum von n und m sein sollte. Die Vereinigung zweier Mengen läßt sich sogar in konstanter Zeit berechnen.

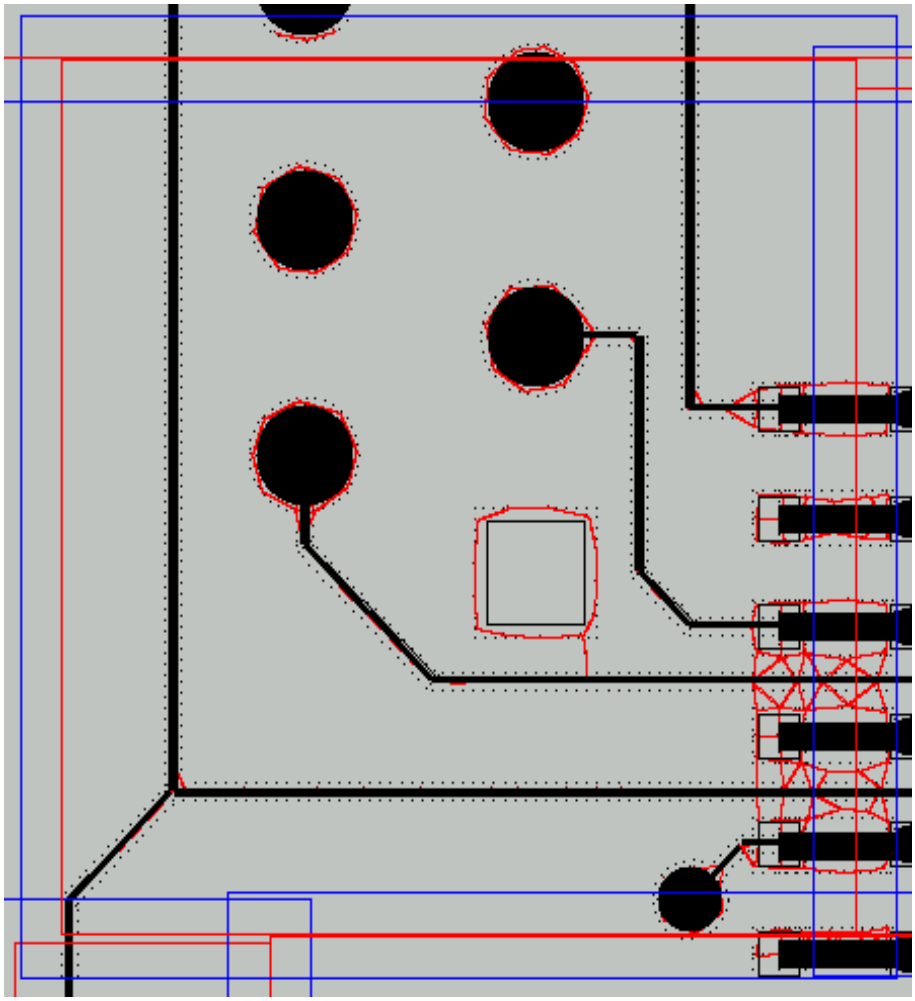

***Abb. 9: Kachel während der Berechnung***

# 4 Ergebnisse

Durch die Verwendung eines neuronalen Ansatzes wurde untersucht, ob das Problem der korrekten Identifizierung von Leiterbahnnetzen aus unsauberen Daten möglich ist, bzw. sich sogar besser und robuster gegenüber herkömmlichen Methoden verhält. Eine mögliche Vorgehensweise zum Filtern unsauberer Daten ist die Betrachtung eines zusätzlichen Randbereichs um jedes geometrische Objekt. Schneiden sich diese Randbereiche, so handelt es sich an der Schnittstelle möglicherweise um eine unsaubere Stelle in den Repräsentationsdaten. Die Korrektheit dieser Methode ist jedoch entscheidend davon abhängig, welche Form der hinzugenommene Randbereich zum Objekt hinzufügt.

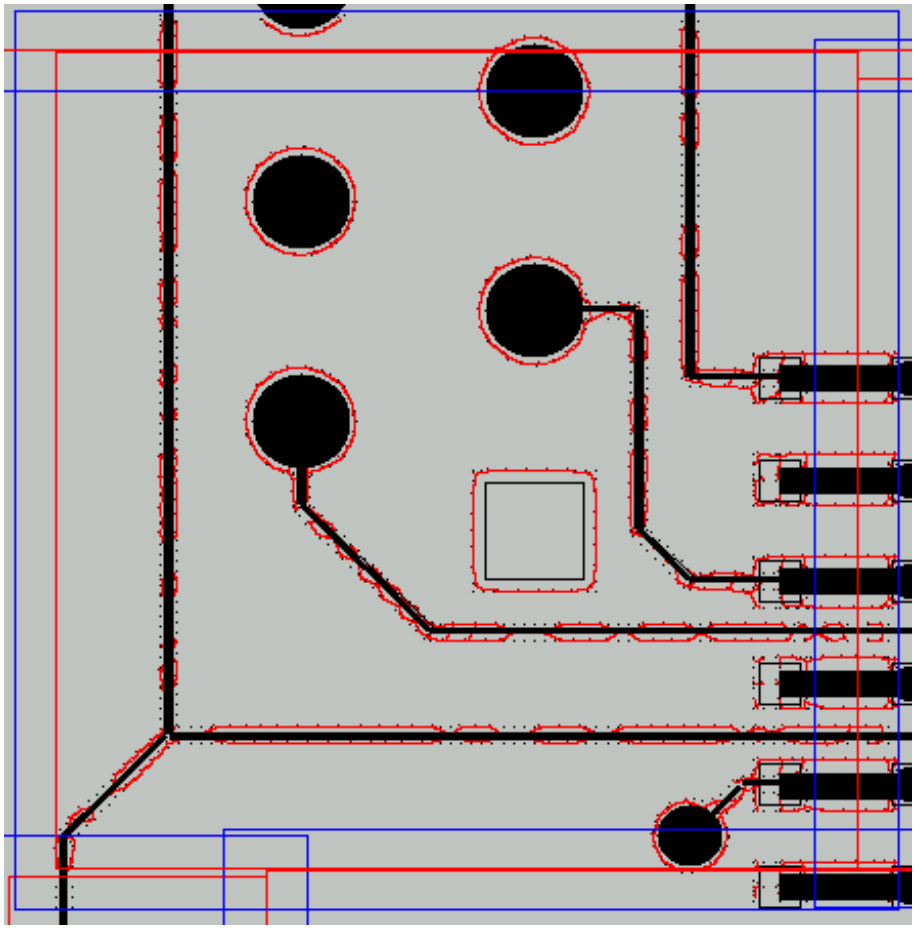

***Abb. 10: Vollständig berechnete Kachel***

Der Vorteil der neuronalen Methode besteht darin, daß diese Form nicht weiter interessiert, da allein durch die höhere Punktdichte in diesem Bereich eine Verbindung zum Nachbarobjekt mit höherer Wahrscheinlichkeit zustande kommt als in anderen Bereichen.
Ein Spezialfall dieser höheren Dichte sind die weiter oben angesprochenen *Drawn Pads*. Typisch für das Erkennen dieser Pads mittels der beschriebenen neuronalen Struktur ist eine erhöhte Verbindungsdichte in diesem Bereich. Deshalb können in einem Nachbearbeitungsschritt diese Pads sehr leicht erkannt werden. Dazu muß lediglich die Anzahl der Verbindungen, die von einem GOB zu einem Nachbarobjekt bestehen, gezählt werden und diese Zahl relativ zu den insgesamt auf diesem Objekt verteilten Testpunkten betrachtet werden. Wird dabei ein bestimmter Prozentsatz überschritten, handelt es sich um ein *Drawn Pad* und

die adjazenten Objekte können zu einem Objekt vereinigt werden. Ein Beispiel dafür zeigt Abb. 11.
In den Abbildungen sind jeweils die Kachelgrenzen (rot) mit zugehörendem Randbereich (blau) zu sehen. Damit die Neuronenverbindungen deutlicher zu sehen sind, wurden die geometrischen Objekte geringfügig kleiner gezeichnet als sie zur Berechnung herangezogen werden. Bei Leiterbahnzügen verlaufen die Neuronenverbindungen oftmals in der Mitte der Linie und sind nicht immer genau erkennbar.

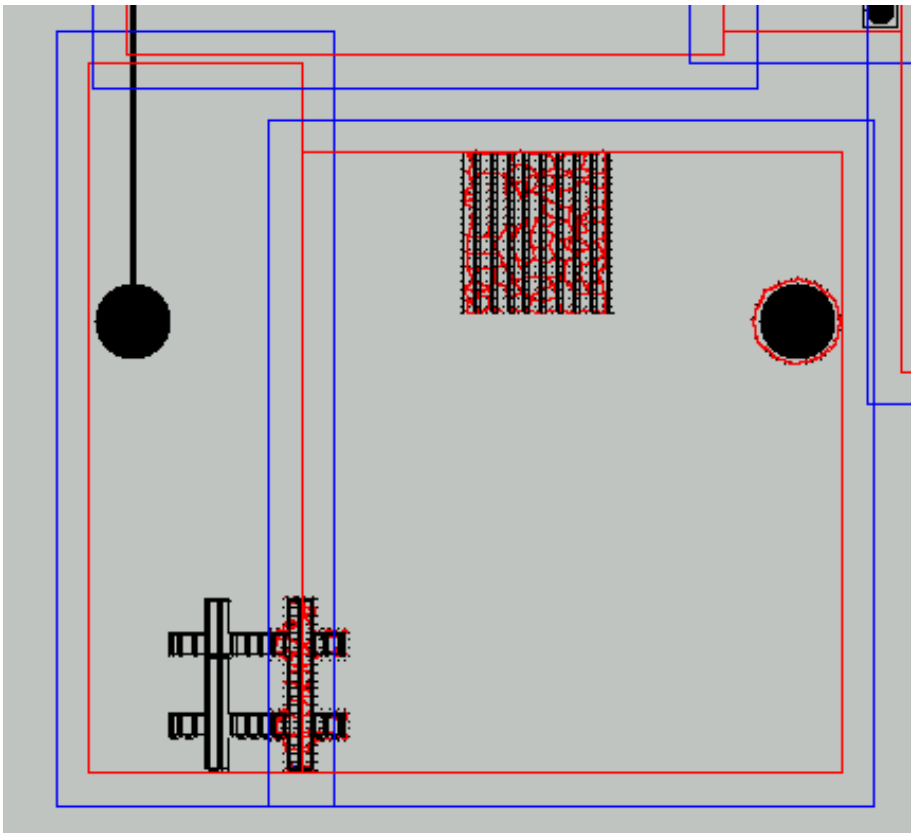

***Abb. 11: Erkennen von Drawn Pads***

## 5 Ausblick

Praktische Untersuchungen an realen Leiterplattendaten haben gezeigt, daß eine Interpretation mittels des vorgestellten Verfahrens möglich ist und Vorteile hinsichtlich der Korrektur fehlerhafter Daten gegenüber standardisierten Methoden bietet. Die Rechenzeit des vorgestellten Verfahrens läßt sich an vielen Stellen durch die Wahl von besseren Datenstrukturen weiter optimieren. So würde sich durch die Verwendung von Voronoi-Diagrammen während der UFDCS-Phase die aufwendige Nachbarschaftssuche entscheidend verkürzen. Das würde es ermöglichen, die Kachelung der Gesamtplatine gröber zu belassen.
Durch die verteilte Struktur der Testpunkte auf einzelne Kacheln ist jederzeit eine Parallelisierung des Verfahrens völlig problemlos möglich und praktisch ohne Änderungen des Algorithmus zu erreichen.
Die Autoren möchten sich bei Prof. Dr. P. Tran-Gia und Kurt Tutschku für ihre Hinweise und Anregungen bedanken. Ebenfalls sei die gute Zusammenarbeit mit den Mitarbeitern der Firma ATG hervorzuheben.

## Literatur


[1] **T. Kohonen**, *Self-Organization and Associative Memory*, Springer-Verlag, New York,1984.

[2] **J. Bruske, G. Sommer**, Dynamic Cell Structure Learns Perfectly Topology Preserving Map, Neural Computation, MIT, 1995.

[3] **B. Fritzke**, *Wachsende Zellstrukturen – Ein selbstorganisierendes neuronales Netzwerk*, Arbeitsbericht des IMMD, Universität Erlangen-Nürnberg, Erlangen, 1992.

[4] **T. Ottmann, P. Widmayer**, *Algorithmen und Datenstrukturen*, Spektrum Akademischer Verlag, 1996.

[5] **The Gerber Scientific Instrument Company**, *Gerber Format Specs*, South Windsor, CT, USA ,1983.

[6] **D. Stoyan, H. Stoyan**, *Fraktale-Formen-Punktfelder: Methoden der Geometrie-Statistik*, Akademie Verlag, Berlin, 1992.