

Implementierung eines Single Pass Connected Component Labeling Algorithmus zur Detektion von leuchtenden Objekten in Nachtszenen im Automotive Umfeld

Steffen Jaeckel, Axel Sikora, Wolfgang Rölling

Zitiervorschlag im APA Stil:

Jaeckel, S., Sikora, A., & Rölling, W. (2010). Implementierung eines Single Pass Connected Component Labeling Algorithmus zur Detektion von leuchtenden Objekten in Nachtszenen im Automotive Umfeld. *Tagungsband zum Workshop der Multiprojekt-Chip-Gruppe Baden-Württemberg, Göppingen, 5. Februar 2010, 43, 67–72.* <https://nbn-resolving.org/urn:nbn:de:bsz:ofb1-opus4-60221>

Abstract

Der Connected Component Labeling (CCL) Algorithmus wird in vielen bildverarbeitenden Systemen eingesetzt, die zur Objekterkennung genutzt werden. Er wird als Basisalgorithmus verwendet, um bestimmte Teile eines Bildes als ein zusammenhängendes Objekt zu identifizieren und somit die weitere Verarbeitung zu vereinfachen. Die ursprüngliche Definition des Connected Component Labeling Algorithmus benötigt zwei Durchläufe durch das Bild, wobei im ersten Durchlauf das Labeling durchgeführt wird. In einem zweiten Durchlauf werden sich berührende Labels zusammengeführt. In diesem Beitrag wird ein Single-Pass-CCL-Algorithmus zur Detektion von leuchtenden Objekten im Automotive Umfeld vorgestellt, der in vielen Fällen zufriedenstellende Ergebnisse liefert. Der Algorithmus wurde auf einem Altera FPGA implementiert und in die Systemumgebung einer Bildererkennung aus dem Automotive Umfeld integriert. Ebenfalls wurde eine web-basierte Testumgebung erstellt. Diese Implementierung dient auch als Framework für die Entwicklung und Umsetzung verbesserter Algorithmen.

Nutzungsbedingungen

Dieses Dokument wird unter diesen Bedingungen zur Verfügung gestellt:
Urheberrechtlich geschützt
Für weitere Informationen siehe:
<https://rightsstatements.org/page/InC/1.0/>

Kontakt

Hochschule Offenburg | Bibliothek
Badstraße 24
77652 Offenburg
Telefon: (0781) 205-240
E-Mail: bibliothek@hs-offenburg.de
www.hs-offenburg.de/bibliothek

Implementierung eines Single Pass Connected Component Labeling Algorithmus zur Detektion von leuchtenden Objekten in Nachtszenen im Automotive Umfeld

Dipl.-Inform. (FH) Steffen Jaeckel, Prof. Dr.-Ing. Axel Sikora und Prof. Dr. Wolfgang Rülling
SIZ Embedded Design und Networking, Poststr. 35, 79423 Heitersheim
Tel. +49-7634-6949341, Mail jaeckel@stzedn.de

Der **Connected Component Labeling (CCL) Algorithmus** wird in vielen bildverarbeitenden Systemen eingesetzt, die zur Objekterkennung genutzt werden. Er wird als **Basisalgorithmus** verwendet, um bestimmte Teile eines Bildes als ein zusammenhängendes Objekt zu identifizieren und somit die weitere Verarbeitung zu vereinfachen.

Die ursprüngliche Definition des **Connected Component Labeling Algorithmus** benötigt zwei Durchläufe durch das Bild, wobei im ersten Durchlauf das Labeling durchgeführt wird. In einem zweiten Durchlauf werden sich berührende Labels zusammengeführt.

In diesem Beitrag wird ein **Single-Pass-CCL-Algorithmus** zur Detektion von leuchtenden Objekten im Automotive Umfeld vorgestellt, der in vielen Fällen zufriedenstellende Ergebnisse liefert. Der Algorithmus wurde auf einem **Altera FPGA** implementiert und in die Systemumgebung einer **Bildererkennung** aus dem Automotive Umfeld integriert. Ebenfalls wurde eine **web-basierte Testumgebung** erstellt. Diese Implementierung dient auch als **Framework** für die Entwicklung und Umsetzung verbesserter Algorithmen.

1. Motivation

Immer mehr Automobile werden mit Kameras ausgestattet. Diese können mit einem oder mehreren Bildaufnehmern zahlreiche Funktionen übernehmen. Hierzu zählen Sicherheits-, bzw. Warnfunktionen, wie z.B. Lane Keep Assist (LKA), Lane Change Assist (LCA), Verkehrszeichenerkennung, Erkennung von Fußgängern oder anderen verletzlichen Straßenbenutzern (Vulnerable Road Users, VRUs) und Komfortfunktionen, wie z.B. ein automatisches Fern- und Abblendlicht oder eine Nachtsichtunterstützung.

Hierbei ist davon auszugehen, dass kamerabasierte Systeme in wenigen Jahren in praktisch allen Neuwagen zu finden sein werden, auch in

kostengünstigeren Kleinwagen. Deswegen ist es wichtig, dass die Basisfunktionen in kostengünstiger Hardware umgesetzt werden können und nicht schnelle und teure Mikroprozessoren hierfür verwendet werden.

Eine Basisaufgabe ist neben der Vorverarbeitung und Vorfilterung die Erkennung zusammenhängender Objekte, die dann als Region of Interest (ROI) näher untersucht werden kann.

2. Architektur objekterkennender Systeme

2.1. Überblick

Objekterkennende Systeme gliedern sich in die bildverarbeitenden Systeme ein. Es handelt sich dabei um Systeme die ein aufgenommenes Bild vorverarbeiten und unter dem Einsatz bildverarbeitender Algorithmen bestimmte Merkmale extrahieren.

Die Aufgaben in einem objekterkennenden System werden in mehreren Schritten durchgeführt (vgl. Bild 1).

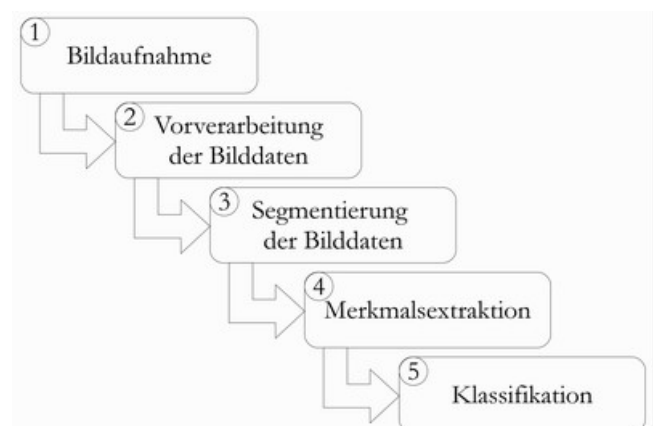


Bild 1 Die einzelnen Ablaufschritte bei der Objekterkennung
Dieser Beitrag beschäftigt sich hauptsächlich mit dem Schritt 4. Es wird jedoch auch kurz auf die

Bearbeitungsschritte 2 und 3 eingegangen da diese Schritte die Grundlage für eine erfolgreiche Merkmalsextraktion legen.

2.2. Vorverarbeitung

Der Schritt der Vorverarbeitung dient dazu, Fehler in den Bilddaten zu korrigieren und die Bildqualität zu verbessern.

$$U(x, y) = \begin{pmatrix} x-1, y-1 & x, y-1 & x+1, y-1 \\ x-1, y & x, y & x+1, y \\ x-1, y+1 & x, y+1 & x+1, y+1 \end{pmatrix} \quad (1)$$

In der Bildverarbeitung werden vielfach lokale Operationen ausgeführt. Hierbei werden neben dem Bildpunkt an Position (x, y) auch die direkt benachbarten Bildpunkte, die lokale Umgebung U , mit einbezogen, vgl. Gl. (1).

$$m = \frac{1}{16} \cdot \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} * U(x, y) \quad (2)$$

Zur Verbesserung der Eingangsbilddaten wird ein Gauß Tiefpassfilter angewendet, vgl. Gl. (2). Es existieren einige Tiefpassfilter, denen ein besseres Glättungsverhalten nachgesagt wird, wie der Median oder der Olympic Filter. Die Implementierung dieser Filter auf einem FPGA ist allerdings wesentlich komplexer, weshalb deren Verwendung außen vor bleibt.

2.3. Segmentierung

Bei der Segmentierung wird das gesamte Bild in einzelne Bereiche, bzw. Objekte mit gleichen Eigenschaften unterteilt. Dabei muss darauf geachtet werden, dass das richtige Segmentierungsverfahren in der benötigten Auflösung verwendet wird. Einerseits würde ein zu detailreiches Bild die nächsten Schritte der objekterkennenden Operationen unnötig aufblähen. Auf der anderen Seite dürfen aber die relevanten Bildinformationen durch eine zu geringe Auflösung nicht verloren gehen.

Laut [5] gibt es mehrere verschiedene Klassen von Segmentierungsverfahren. Dazu zählen

- pixelorientierte Verfahren
- kantenbasierte Verfahren
- regionenbasierte Verfahren
- modellbasierte Verfahren

In Bild 2 wird das pixelorientierte Segmentierungsverfahren der adaptiven Schwellwertoperation demonstriert. Der Ausdruck adaptive Schwellwertoperation bedeutet dabei, dass auf Grund der entstandenen Ergebnisse einer ersten Schwellwertoperation die Grenze des Schwellwertes angepasst

(adaptiert) werden kann, um das Ergebnis zu beeinflussen.

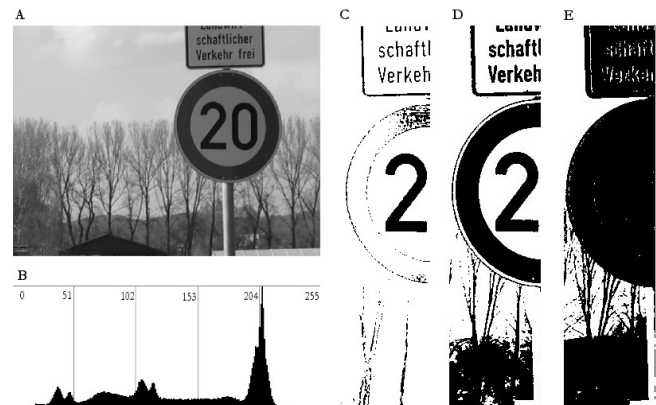


Bild 2 Segmentierung eines Bildes bei ungleichmäßigem Hintergrund: A Originalbild von Fb78; B Histogramm; C bis E Ausschnitt des Bildes nach Schwellwertoperation mit einem globalen Schwellwert von 40, 95 bzw. 130

Bei der Segmentierung mit einem Schwellwert wird aus einem Farb-, bzw. Graustufenbild ein Binärbild extrahiert. An Stellen, an denen der Wert des Bildpunktes unterhalb der Grenze des Schwellwerts liegt, wird der Bildpunkt zu Hintergrund, an Stellen oberhalb der Grenze wird der Punkt zu Vordergrund. Wie in Bild 2 klar erkenntlich ist, entstehen merkliche Unterschiede in den Ergebnissen in Abhängigkeit von der Wahl des Schwellwertes.

2.4. Merkmalsextraktion

Nach [4] werden in diesem Schritt die für die Klassifikation bedeutsamen Merkmale extrahiert und in einer Liste abgelegt. Dies ist der schwierigste Schritt in der Objekterkennung beim computerbasierten Sehen, da für die Merkmale "scharfe" Kriterien angewendet werden. Im Gegensatz dazu werden bei der menschlichen Merkmalsextraktion "weiche" Kriterien angewendet, was zu einer höheren Robustheit in Bezug auf die Falscherkennung von Objekten führt. Für die Merkmalsextraktion stehen mehrere Ansätze zur Verfügung. So lassen sich Ansätze basierend auf Kanten und Ecken realisieren, wie in [3] beschrieben. Laut [6] wäre auch ein Ansatz der auf skalierungsinvarianten Merkmalen basierend zielführend. Man kann auch den Ansatz einer Hauptkomponentenanalyse nach [8] verfolgen. Der vorgestellte Connected Component Labeling (CCL) Algorithmus unterscheidet sich hauptsächlich durch seine Einfachheit und sequenzielle Ausführbarkeit von den meisten anderen Ansätzen.

Der CCL-Algorithmus wird in vielen bildverarbeitenden Systemen eingesetzt, die zur Objekterkennung genutzt werden. Er wird als Basisalgorithmus verwendet, um bestimmte Teile eines Bildes als ein

zusammenhängendes Objekt, eine so genannte Zusammenhangskomponente (ZHK), zu identifizieren und somit die weitere Verarbeitung zu vereinfachen.

Die ursprüngliche Definition des Connected Component Labeling Algorithmus ist in [10] beschrieben und benötigt zwei Durchläufe durch das Bild, wobei im ersten Durchlauf das Labeling durchgeführt wird, bei dem allen Bildpunkten eines Bildes ein Label zugewiesen wird. Sich berührende Labels werden mit einer modifizierten Variante des in [9] vorgestellten *UNION* Operator verkettet. Im zweiten Durchlauf wird mit Hilfe des *FIND* Operators aus [9] für jedes Label die Wurzel gesucht und ihm zugewiesen. Nach dem zweiten Durchlauf besitzen sich berührende Bildpunkte ein eindeutiges Label und können für weitere Schritte getrennt betrachtet werden.

In der Implementierung des CCL Algorithmus gibt es grundsätzlich die Möglichkeit, unterschiedliche Arten der Connectivity/Verbindung/Nachbarschaft einzubeziehen. Eine Möglichkeit ist die Benutzung der 4-connectivity, bzw. Vierernachbarschaft, wobei die vier direkt anliegenden Bildpunkte zur Betrachtung hinzugezogen werden. Außerdem gibt es noch die 8-connectivity, bzw. Achternachbarschaft, bei der alle acht umliegenden Bildpunkte betrachtet werden. Die Auswahl der verwendeten Art der Nachbarschaft hängt von der Komplexität der Eingangsdaten und dem angestrebten Ergebnis ab und variiert je nach Einsatzgebiet und der verfügbaren Leistung des bildverarbeitenden Systems.

Grundsätzlich lässt sich der Connected Component Labeling Algorithmus laut [1] und [11] auf jegliche Datenstrukturen anwenden. So wäre es denkbar, eine Version des CCL-Algorithmus zur Vorbereitung der Merkmalsextraktion aus den rohen Bilddaten einzusetzen und eine abgewandelte Form des CCL-Algorithmus auch in späteren Arbeitsschritten, wie der Klassifikation, zu verwenden.

2.5. Klassifikation

Die Klassifikation in der Bildverarbeitung reiht sich ein in die Methoden der Mustererkennung (vgl. Schritt 4 in Bild 1). Dabei ist das Ziel der Klassifikation, den durch die Merkmalsextraktion gefundenen Objekten Bedeutungen zuzuweisen.

Die Klassifikation umfasst laut [5] zwei grundlegende Aufgaben.

Es müssen Beziehungen zwischen den Bildeigenschaften und den Objektklassen so detailliert wie möglich herausgearbeitet werden.

Es muss der optimale Satz an Bildeigenschaften herausgefunden werden, so dass mit einer möglichst

einfachen Klassifizierungsmethode die Objekte in die verschiedenen Klassen eingeteilt werden können.

Dem Klassifikationsverfahren muss bereits vor der Ausführung bekannt sein, welche Bedeutungen denn erkannt werden sollen. Dies kann entweder statisch geschehen, durch eine feste Vorgabe der Merkmale einer Bedeutung oder man die Merkmale auch durch Verfahren des Maschinellen Lernens ermitteln. Tiefere Einblicke in Klassifikationsverfahren können bezogen werden aus den Werken [12], [5] und [2].

3. Connected Component Labeling

3.1. Klassischer Algorithmus

Der Connected Component Labeling Algorithmus wird auf einem binarisierten Bild ausgeführt, das üblicherweise aus einer Schwellwertbildung eines Graustufenbildes gewonnen wurde. Dabei wird das binarisierte Bild pixelweise beginnend von links oben, Zeile für Zeile durchlaufen und seine Vierernachbarschaft betrachtet. Aufgrund der zeilenweisen Betrachtung des Gesamtbildes müssen bloß das nördliche und das westliche Label betrachtet werden, so dass am Ende des Bildes jedes Pixel, das nicht Hintergrund ist, ein Label besitzt.

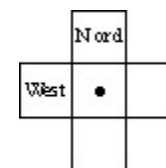


Bild 3 Vierernachbarschaft

Für den Fall, dass sich mehrere Labels berühren, wird mittels des Union-Find-Algorithmus ein passendes Label ausgewählt und zugewiesen. Dazu wurde die in [9] vorgestellte Implementierung dahingehend modifiziert, dass nicht die Größe der Zusammenhangskomponente (ZHK) als Entscheidung genommen wurde, welche ZHK denn zugewiesen wird, sondern immer die kleinere ZHK ausgewählt wird.

Mit dem Abschluss der Bearbeitung des Bildes wurde jedem Pixel, das nicht als Hintergrund gilt, ein Label vergeben. Dabei können auch komplexe Beziehungen der ZHKs entstehen, wenn verschiedene Regionen zum Beispiel V-förmig zusammenlaufen oder eine Art Schneckenform besitzen. Solche Fälle müssen daher speziell behandelt werden, weshalb der Algorithmus zwei Durchläufe benötigt.

Im ersten Durchlauf werden alle Pixel betrachtet. Wenn ein Pixel kein Hintergrund ist, wird ihm ein bestimmtes Label zugewiesen, das in eine Tabelle eingetragen wird, die den Dimensionen des zu

bearbeitenden Bildes entspricht. In einer zweiten Tabelle werden nebenher die Verkettungen, die aus dem Union-Find Algorithmus entstehen, gespeichert.

```
function find(a) returns int
  result = a
  do {
    -- search parent of a
    result = result.parent
  } while (result != result.parent)
  -- Path compression
  while(a != result)
  {
    help = a.parent
    a.parent = result
    a = help
  }
  return result

function union(a, b) returns int
  aRoot = find(a)
  bRoot = find(b)
  if(aRoot < bRoot)
    bRoot.parent = aRoot
  else
    aRoot.parent = bRoot
  return bRoot
```

Listing 1 Modifizierter Union-Find Algorithmus

3.2. Single Pass Union-Find Ansatz

Angelehnt an den klassischen Ansatz wurde mit Hilfe von Prof. Dr. Wolfgang Rülling eine modifizierte Variante des CCL Algorithmus entworfen. Diese Variante nutzt gewisse Eigenschaften aus, die sich dadurch ergeben, dass das Bild bloß einmal durchlaufen wird.

- Bei der Abarbeitung des Bildes von oben nach unten entsteht eine Verkettungsstruktur, in der sämtliche Kanten von rechts nach links verlaufen.
- Wegen der Abarbeitung der Zeilen von links nach rechts müssen maximal Ketten der Länge 2 durchlaufen werden.

Der Algorithmus setzt die gleiche Union-Find Implementierung ein, die in Listing 1 vorgestellt wurde. Bei der Ausführung stehen die Bildpunkte bzw. ihre Repräsentanten der "alten Zeile" sowie die der "neuen Zeile" zur Verfügung. Die Zeilen werden von links nach rechts durchlaufen und an jeder Position

müssen folgende drei Operationen ausgeführt werden.

1. Der Bildpunkt in der neuen Zeile verweist zunächst auf sich selbst. Ist der linke Nachbar P_{links} gesetzt, soll der aktuelle Bildpunkt P_{neu} auf den Repräsentanten des linken Nachbarn verweisen: $UNION(P_{links}, P_{neu})$
2. In der alten Zeile wird für jeden Bildpunkt P_{alt} die $FIND()$ -Operation ausgeführt. Dabei wird der Repräsentant des derzeit bearbeiteten Bildpunkts P_{alt} ermittelt und die Kette in der alten Zeile komprimiert. Hier werden maximal Ketten der Länge 2 durchlaufen und es entstehen Ketten der Länge 1.
3. Wenn der Bildpunkt P_{neu} einen Nachbarn P_{alt} in der darüber liegenden alten Zeile besitzt, muss außerdem eine $UNION$ Verknüpfung von P_{alt} und P_{neu} erfolgen, $UNION(P_{alt}, P_{neu})$. Dabei werden nicht die wirklichen Bildpunkte verwendet, sondern ihre Repräsentanten die in den Schritten 1 und 2 ermittelt wurden. Dabei muss beachtet werden:

Wenn der Repräsentant von P_{alt} noch in der alten Zeile liegt, muss man den Repräsentanten von P_{alt} auf den Repräsentanten von P_{neu} (in die neue Zeile) zeigen lassen.

Wenn der Repräsentant von P_{alt} bereits in der neuen Zeile liegt, muss man den rechten Repräsentanten auf den linken zeigen lassen.

Außerdem muss beachtet werden

- dass man in der Implementierung erkennen muss, ob der Repräsentant eines Bildpunkts in der alten oder in der neuen Zeile liegt.
- dass man, sobald die neue Zeile vollständig durchlaufen wurde, in der alten Zeile nach existenten Repräsentanten suchen muss, die in die alte Zeile zeigen. Diese Komponenten können ausgegeben werden, da sie ihren Repräsentanten in der alten Zeile haben und somit als abgeschlossen gelten.

3.3. Schneller Ansatz

Der derzeit implementierte Algorithmus ist eine Modifikation des klassischen CCL-Algorithmus. Das Hauptaugenmerk beim Entwurf dieses Algorithmus lag darauf in konstanter Laufzeit eine Extraktion der gewünschten Merkmale zu erreichen. Dieser Algorithmus wurde exakt beschrieben und in *VHDL* implementiert. Er arbeitet wie der Algorithmus aus Kap. 3.1 in einer Vierernachbarschaft. Dieser Ansatz setzt im Gegensatz zum Single Pass Union-Find Ansatz aus Kap. 3.2 keine Verkettungen der

Bildpunkte mittels Union-Find ein, sondern erlaubt maximal eine Verkettung einer Komponente pro Zeile. Durch diese Anpassung kann die Implementierung ein konstante Durchlaufzeit und eine hohe Bearbeitungsrate von 1 Bildpunkt pro Takt erreichen.

4. Derzeitige Implementierung

4.1. Überblick

Im Laufe der Arbeiten wurde ein bildverarbeitendes System implementiert, das als Vorverarbeitungsstufe in einem objekterkennenden System dient. Dabei wurden die Punkte 2 bis 4 aus Bild 1 in ein System auf Basis eines Altera FPGA implementiert, s. Kap. 4.2.

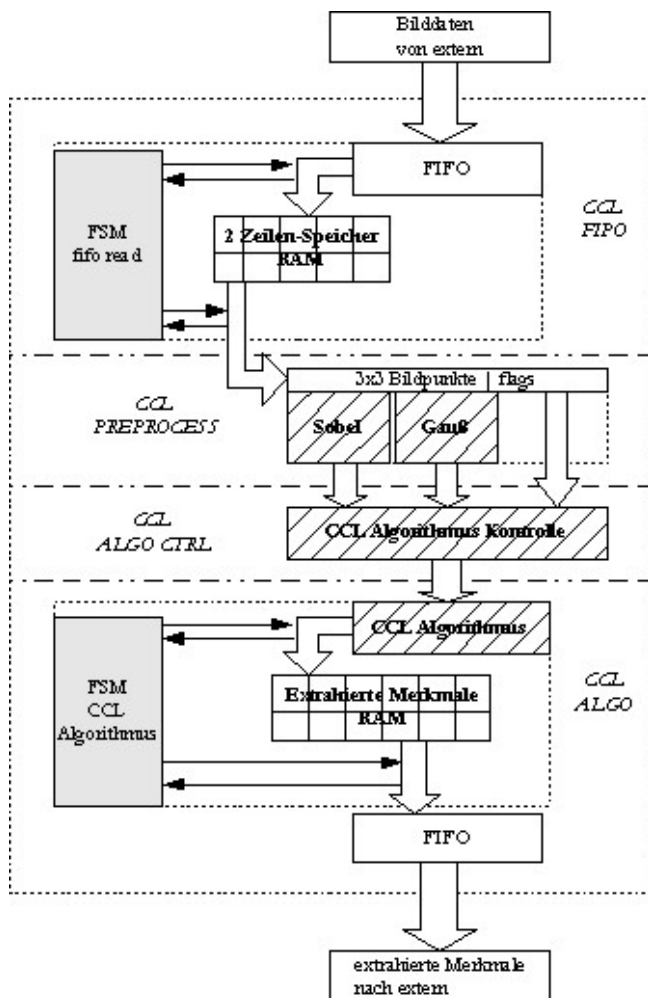


Bild 4 Überblick der implementierten Funktionalität

Die Vorverarbeitung der Bilddaten, vgl. Punkt 2 in Bild 1, wurde in den Modulen *CCL FIPO* und *CCL PREPROCESS*, vgl. Bild 4, implementiert. Das *CCL*

FIPO Modul erzeugt eine 3x3 Pixel große Matrix. Diese Matrix wird im *CCL PREPROCESS* Modul zur Vorverarbeitung der Bilddaten, wie in Kap. 2.2 gezeigt, benötigt.

Die Segmentierung der Bilddaten, vgl. Punkt 3 in Bild 1, ist im *CCL ALGO CTRL* Modul implementiert. Dabei wird auf den Tiefpass-gefilterten Originalpixel, wie in Kap. 2.3 gezeigt, eine adaptive Schwellwertoperation angewendet, was zu einer Binarisierung des Eingangsbilds führt.

Dieses binarisierte Bild wird als Eingang zur Merkmalsextraktion im *CCL ALGO* genutzt.

4.2. Implementierungsdetails

Das Gesamtsystem wurde dabei auf folgende Rahmendaten ausgelegt:

- Bildformat: 640x480 Pixel
- Pixeltiefe: 8 Bit
- Max. Labels: 64
- Merkmale:
 - Anzahl Pixel
 - Max.- & Durchschnitts-Grauwert
 - Anzahl Kantenpixel
 - Durchschnitts-Gradientenwert
 - Schwerpunkt
 - Bounding Box

Dabei sind die Parameter „Bildformat“, „Pixeltiefe“ und „maximale Anzahl an Labels“ zur Kompilierzeit einstellbar.

Das System kann selbst für einen Altera Cyclone3 Low-Cost FPGA kompiliert und synthetisiert werden. Eine Begrenzung der verwendbaren Taktfrequenz entsteht nur durch die Grenzen der eingesetzten Hardware. In Tab. 1 wird der Ressourcenverbrauch des implementierten bildverarbeitenden Systems gezeigt.

Tab. 1 Ressourcenverbrauch absolut

Modul	Logic Cells	Dedicated Logic Registers	Block RAMs
<i>FIPO</i>	521	326	2
<i>PREPROC</i>	122	74	0
<i>ALGOCTRL</i>	31	24	0
<i>ALGO</i>	647	239	3

Die Daten aus Tab. 1 beziehen sich auf die verwendete Logik für den *CCL Algorithmus* und die

Extraktion des Merkmals der Fläche (Anzahl der Bildpunkte) in einer ZHK. Die zwei FIFOs am Modul-Eingang, bzw. Ausgang (vgl. Bild 4) gehen nicht in den gezeigten Ressourcenverbrauch ein. Sie sind jedoch nötig, um eine Integration in das Gesamtsystem zu ermöglichen.

5. Ausblick

Die Implementierung dieses ersten CCL-Blocks diene zum Aufbau der gesamten Infrastruktur. In diesen Rahmen konnte mittlerweile ein erweiterter CCL-Algorithmus [7] integriert werden, der eine größere Anzahl von benachbarten Zellen in die Zusammenhangsbetrachtung einbezieht und auch noch mehr Sonderfälle berücksichtigt.

Über diesen wird gesondert berichtet.

6. Literaturverzeichnis

[1] M. B. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *J. ACM*, 39 (2): 253–280, 1992. ISSN 0004-5411.

[2] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Prentice Hall, August 2007. ISBN 013168728X.

[3] C. Harris and M. Stephens. A combined corner and edge detector. In *Proc. 4th Alvey Vision Conference*, pages 147–152, 1988.

[4] I. Jahr. *Lexikon der industriellen Bildverarbeitung*. Spurbuchverlag, 2003. ISBN 3887782879.

[5] B. Jähne. *Digitale Bildverarbeitung (German Edition)*. Springer, 2005. ISBN 3540249990.

[6] D. Lowe. Object recognition from local scale-invariant features. In *ICCV99*, pages 1150–1157, 1999.

[7] N. Ma, D. G. Bailey, and C. T. Johnston. Optimised single pass connected components analysis. In *International Conference on Field-Programmable Technology*, pages 185–192, 2008.

[8] K. Pearson. On lines and planes of closest fit to a system of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, series 6, 2: 559–572, 1901.

[9] W. Rülling. *Effiziente Algorithmen und Datenstrukturen*. Vorlesungsskript an der Hochschule Furtwangen University, 2008.

[10] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital image processing. *Journal of the ACM*, 1966.

[11] H. Samet and M. Tamminen. Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10 (4): 579–586, 1988.

[12] R. Schmid. *Industrielle Bildverarbeitung*. Vieweg Verlagsgesellschaft, 2002. ISBN 3528049456.