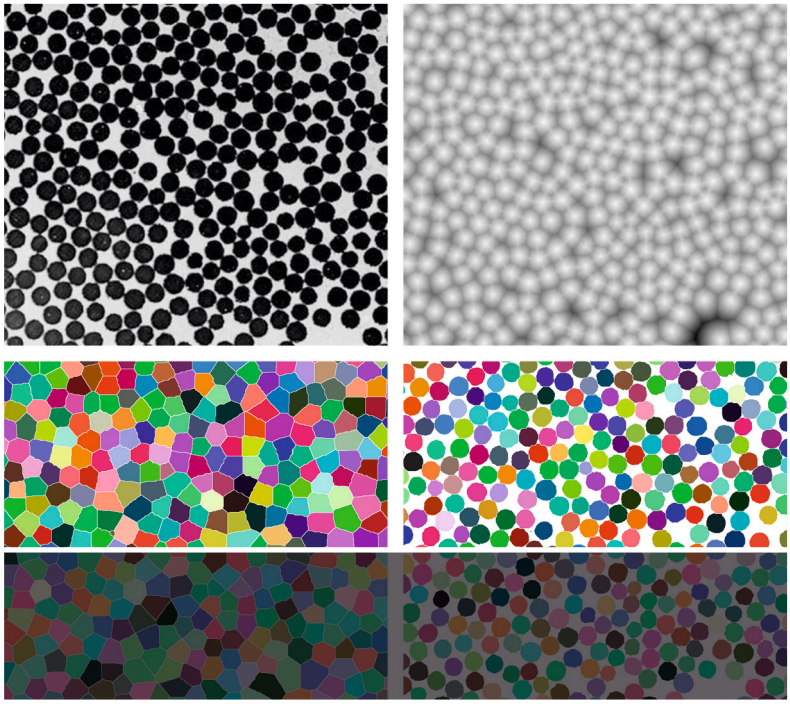
Joachim Ohser

PLUS.HANSER-FACHBUCH.DE

Angewandte Bildverarbeitung und Bildanalyse

Methoden, Konzepte und Algorithmen in der Optotechnik, optischen Messtechnik und industriellen Qualitätskontrolle



2., überarbeitete und erweiterte Auflage

HANSER

Ohser

Angewandte Bildverarbeitung und Bildanalyse

Bleiben Sie auf dem Laufenden!

Hanser Newsletter informieren Sie regelmäßig über neue Bücher und Termine aus den verschiedenen Bereichen der Technik. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter

www.hanser-fachbuch.de/newsletter

Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial. Geben Sie dazu einfach diesen Code ein:

plus-9sgnf-nmuem

plus.hanser-fachbuch.de

Joachim Ohser

Angewandte Bildverarbeitung und Bildanalyse

Methoden, Konzepte und Algorithmen in der Optotechnik, optischen Messtechnik und industriellen Qualitätskontrolle

2., überarbeitete und erweiterte Auflage

HANSER

Über den Autor:
Prof. Dr. Joachim Ohser lehrte Mathematik und Bildverarbeitung an der Hochschule Darmstadt.

Print-ISBN: 978-3-446-47910-4
E-Book-ISBN: 978-3-446-48055-1

Alle in diesem Werk enthaltenen Informationen, Verfahren und Darstellungen wurden zum Zeitpunkt der Veröffentlichung nach bestem Wissen zusammengestellt. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund sind die im vorliegenden Werk enthaltenen Informationen für Autor:innen, Herausgeber:innen und Verlag mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Autor:innen, Herausgeber:innen und Verlag übernehmen infolgedessen keine Verantwortung und werden keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Weise aus der Benutzung dieser Informationen – oder Teilen davon – entsteht. Ebenso wenig übernehmen Autor:innen, Herausgeber:innen und Verlag die Gewähr dafür, dass die beschriebenen Verfahren usw. frei von Schutzrechten Dritter sind. Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt also auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benützt werden dürften.

Die endgültige Entscheidung über die Eignung der Informationen für die vorgesehene Verwendung in einer bestimmten Anwendung liegt in der alleinigen Verantwortung des Nutzers.

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet unter http://dnb.d-nb.de abrufbar.

Dieses Werk ist urheberrechtlich geschützt.
Alle Rechte, auch die der Übersetzung, des Nachdruckes und der Vervielfältigung des Werkes, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Einwilligung des Verlages in irgendeiner Form (Fotokopie, Mikrofilm oder einem anderen Verfahren), auch nicht für Zwecke der Unterrichtgestaltung – mit Ausnahme der in den §§ 53, 54 UrhG genannten Sonderfälle –, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

© 2024 Carl Hanser Verlag GmbH & Co. KG, München
www.hanser-fachbuch.de
Lektorat: Frank Katzenmayer
Herstellung: Frauke Schafft
Coverkonzept: Marc Müller-Bremer, www.rebranding.de, München
Satz: Joachim Ohser
Druck: CPI Books GmbH, Leck
Printed in Germany

Vorwort

Die digitale Bildverarbeitung ist seit vielen Jahrzehnten ein sich sehr schnell entwickelndes Gebiet. Die Ursachen liegen in der immer größer werdenden Vielfalt der Bildgebungsverfahren, den inzwischen zahllosen Anwendungsgebieten und verfeinerten technischen Möglichkeiten für die Verarbeitung und Analyse der Bilddaten. Zu Letzteren gehören die Parallelisierung [43], das Processing auf einer Graphikkarte (*graphics processing unit*, GPU) und die Verwendung programmierbarer logischer Schaltungen (vor allem *field programmable gate arrays*, FPGA), siehe z. B. [10], wo die Leistungen dieser Techniken verglichen werden. Eine zunehmende Rolle spielt dabei die Programmiertechnik CUDA (*compute unified device architecture*), mit deren Hilfe Algorithmen auf der GPU prozessiert werden kann [68, 256]. Die Beschleunigung von Bildverarbeitungsalgorithmen bietet sich wegen ihrer oft inhärenten Parallelität an und eröffnet damit stets neue Anwendungen. Die verschiedenen Anwendungsgebiete beeinflussen ihrerseits die Bildverarbeitung nicht nur dahingehend, dass sie die Entwicklung neuer Methoden anregen; sie tragen darüber hinaus zur Etablierung neuer Teilgebiete bei, die sich durch ihre Ausrichtungen und Fachbegriffe voneinander unterscheiden. Seit den Anfängen der Bildverarbeitung sind das vor allem die Fotogrammetrie sowie die mikroskopische Bildverarbeitung in Medizin [166, 292], Mineralogie und Werkstofftechnik [192]. Heute ist die Palette der Anwendungen praktisch nicht mehr überschaubar, wobei industrielle Qualitätskontrolle [30, 67], die Überwachung und Steuerung von Prozessen, Anlagen und Fahrzeugen bis hin zur Robotik [31] aus wirtschaftlicher Sicht ein besonderes Gewicht erhalten. Hinzu kommt, dass Bildgebung und Bildverarbeitung in vielen Systemen nicht mehr zu trennen sind. In einigen Fällen ist die Bildverarbeitung sogar integraler Bestandteil der Bildgebung. Man denke nur an moderne Fotografie, Computertomographie (CT) [258, 26, 117, 230], konfokale Laserscanningmikroskopie (CLSM) auf der Basis der Lichtblatttechnologie [122, 266, 160] (*light sheet fluorescense microscopy*, LSFM) oder die Ptychographie [155, 55].

Grundsätzlich lassen sich sehr viele Algorithmen der Verarbeitung und Analyse 2-dimensionaler Bilder auf 3-dimensionale (Volumen-)Bilder übertragen [199], die beispielsweise durch Tomographie oder CLSM erhalten werden. Für 3-dimensionale Bilder von Oberflächen, die mit aktiver Triangulation, mono-, bin- oder multiokularer Stereophotogrammetrie, mittels Lichtfeldkameras, durch Photoklinometrie (*shape from shading*, SFS) oder Autofokussensoren, mit Laufzeitmessungen (*time of flight*, TOF), Interferometrie, Shearographie und Holographie erzeugt werden [231], ist das allerdings nur sehr eingeschränkt möglich. Die Verarbeitung und Analyse solcher Bilder scheint derzeit mehr oder weniger eigenständige Wege zu gehen [39].

Mit diesen Entwicklungen einher geht, dass die Bildverarbeitung zu den zentralen Bestandteilen verschiedenster Fachgebiete gehört, die wiederum maßgeblich zu ihrer Entwicklung beigetragen haben. Dazu zählen insbesondere die Elektro- und Kommunikationstechnik [28, 80, 239] und die Informatik (von Computer-Vision [152] bis *geometric deep learning* [173]), aber auch die Mathematik (vor allem die Diskrete oder Digitale Geometrie [154, 153], die Differentialgeometrie [52], das Gebiet der Partiellen Differentialgleichungen [238, 203] und die Numerik [149]). Die Integralgeometrie mit dem exzellenten Buch von Rolf Schneider [243] (in einer

erweiterten Edition [244]) hat sich zu einer wichtigen Grundlage der Bildanalyse entwickelt, und Büchern über Stochastische Geometrie [175, 61, 245, 57, 139] und zufällige Felder [2, 3] können wertvolle Anregungen zur Analyse von Mikrostrukturen entnommen werden.

Für die Bildverarbeitung und Bildanalyse gibt es zahlreiche eigenständige, von der Art der Bildgebung weitgehend unabhängige und daher allgemein anwendbare Softwarepakete. Dazu gehören das System `OpenCV` (*Open Source Computer Vision Library*) [124], das `C++`-Toolkit `Dlib` [151] mit einem Fokus auf maschinelles Lernen, das in `Java` geschriebene und damit plattformübergreifende System `ImageJ` [242], das System `Halcon` [185] der Firma MVTec Software GmbH, die Software der Fa. Stemmer [4], das System `ToolIP` [88] des Fraunhofer-Instituts für Techno- und Wirtschaftsmathematik, verschiedene Produkte der Firma PixelFerber (Berlin) [211] für die Mikroskopbildverarbeitung, das Modul `LabVIEW Vision` im System `LabVIEW` der National Instruments AG [93] und die Programmbibliothek `Caffe`, die zahlreiche Algorithmen und Deep-Learning-Architekturen für die Klassifikation und Clusteranalyse von Bilddaten enthält [140]. Stärker an die Abbildungstechnik gebunden sind z. B. Softwareprodukte der Firmen Olympus Soft Imaging Solutions GmbH (Münster) und Carl Zeiss MicroImaging GmbH (München). Darüber hinaus sind umfangreiche Pakete für die Bildverarbeitung in die Systeme `Python` der Python Software Foundation [215], `MatLab` der Firma MathWorks [177] und `IDL` (*Interactive Data Language*) [113] integriert. Die Leistungsfähigkeit dieser Systeme, d. h. der Umfang der Algorithmen der Bildverarbeitung und Bildanalyse sowie die Qualität ihrer Implementierung, unterscheidet sich beträchtlich. Dabei scheint es schwerzufallen, ein durchgängiges Konzept hinsichtlich der Ausnutzung der Separabilität, der Berücksichtigung des zugrunde liegenden Gitters, der Wahl der Nachbarschaft der Pixel, der Randbehandlung in Bildern etc. zu wahren [159].

Aus den oben genannten Gründen können im vorliegenden Buch nur wenige Teilgebiete der Bildverarbeitung und Bildanalyse behandelt werden. Das Buch soll vielmehr als ein Lehrbuch verstanden werden, in dem eine Einführung in dieses sehr große Gebiet gegeben wird. Im Vordergrund stehen klassische Methoden der 2-dimensionalen Bildverarbeitung, wobei unter Wahrung der allgemeinen Verständlichkeit des Textes neuere Sichtweisen präsentiert werden. In nur wenigen Abschnitten werden mathematische Kenntnisse vorausgesetzt, die über ein Grundstudium hinausgehen. In didaktischer (und teilweise auch inhaltlicher) Hinsicht soll an den exzellenten Klassiker über morphologische Bildverarbeitung von Jean Serra [250], dem ehemaligen Direktor des Centre de Morphologie Mathématitque in Fontainebleau, angeknüpft werden. Dem Lehrbuchcharakter wird durch zahlreiche, zum überwiegenden Teil sehr leicht nachvollziehbare Beispiele Rechnung getragen. Ergänzende Übungsaufgaben und deren im Anhang präsentierte Lösungen sollen es dem Leser erleichtern, sich in das Stoffgebiet einzuarbeiten. Außerdem werden einige Algorithmen in Form von Quellcode präsentiert, um dazu anzuregen, selbst zu programmieren und eigene Methoden zu implementieren. Zum vertiefenden Studium wird auf das bereits in mehrfacher Auflage erschienene Buch von Bernd Jähne [134, 136] (englischsprachige Fassung [135]), das dreibändige Werk ([48, 49]) von Wilhelm Burger und Mark Burge sowie das Handbuch [229] von John Russ und Brent Neal verwiesen. Eine mehr mathematische Behandlung des Themas ist in dem neueren Buch von Kristian Bredies und Dirk Lorenz [44] sowie in der zweibändigen Ausgabe von Jean-Charles Pinoli [209, 210] zu finden, siehe auch [202] zu Level-Set-Methoden und [282] zur Diffusionsfilterung. Die Bearbeitung von Farbbildern kann in [97] vertiefend nachgelesen werden, und auch zur Verarbeitung und Analyse von 3-dimensionalen (Volumen-)Daten gibt es umfangreiche weiterführende Literatur [171, 188, 267, 199]. Schließlich wird noch auf das Buch von Jürgen Beyerer u. a. [35]

hingewiesen, in dem eine ausgezeichnete Übersicht zur 3-dimensionalen Bildgebung profilierter oder gekrümmter Oberflächen enthalten ist.

Das vorliegende Buch ist wie folgt gegliedert: In Kapitel 1 werden einige Grundlagen der Bildverarbeitung behandelt. Dazu gehören homogene Gitter, auf denen kontinuierliche Bilder gesampelt werden, Pixel und ihre Nachbarschaften sowie der Wechsel des Gitters durch bilineare Interpolation der Pixeldaten. Einige für die Anwendung sehr wichtige Filter wie morphologische Transformationen, lineare und morphologische Filter sowie Rangordnungsfilter sind in Kapitel 2 beschrieben. Bildtransformationen wie das Labeling, die Distanz-, Wasserscheiden-, Radon- und Hough-Transformation werden in Kapitel 3 behandelt. Natürlich zählt auch die Fourier-Transformation zu den Bildtransformationen. Wegen ihrer großen Bedeutung ist ihr und ihren Anwendungen ein eigenständiges Kapitel gewidmet (Kapitel 4), wobei besonderes Augenmerk auf die vielfältigen wechselseitigen Beziehungen zwischen kontinuierlicher und diskreter Fourier-Transformation gelegt wird. Zu den Anwendungen der Fourier-Transformation gehören auch die lineare Filterung und die Korrelationsanalyse via Ortsfrequenzraum, die in Kapitel 5 behandelt werden. Das schließt die bildanalytische Bestimmung der Auto- und Kreuzkorrelationsfunktion von zufälligen Strukturen mit ein. Als weitere Anwendungen der Fourier-Transformation gelten die schnelle Radon-Transformation und ihre Inverse, die tomographische Rekonstruktion (Kapitel 6), wobei Algorithmen wie die gefilterte Rückprojektion zwar ohne Fourier-Transformation auskommen, bei deren Herleitung aber das Projektions-Schnitt-Theorem der Fourier-Transformation verwendet wird. Schließlich wird noch in Kapitel 7 auf einige Aspekte der digitalen Bildanalyse eingegangen, wobei vor allem auf integralgeometrische Ansätze zurückgegriffen wird.

Das Buch richtet sich an Studierende der Elektrotechnik (insbesondere der Automatisierungstechnik und Mechatronik), der Informatik, der Werkstofftechnik sowie der Optotechnik und Bildverarbeitung. Zudem wendet es sich an Entwicklerinnen und Entwickler von Bildverarbeitungssystemen sowie an Ingenieurinnen und Ingenieure, die sich mit dem Einsatz dieser Systeme im industriellen Umfeld befassen. Der Hanser Verlag ermöglicht auf seiner Internetseite plus.hanser-fachbuch.de den Zugriff auf eine kleine C-Bibliothek als ergänzendes Material, das Implementierungen der im Buch beschriebenen Methoden enthält. Außerdem wird dort ein Verzeichnis mit Quellcode zur Verfügung gestellt, der durch Mausklick auf die entsprechende Stelle im eBook geöffnet werden kann. Damit werden diese Methoden noch besser veranschaulicht, und es kann schnell und einfach mit Bilddaten experimentiert werden. Der Zugangscode zu dieser Internetseite ist auf Seite 1 in der Titelei dieses Buches abgedruckt.

Abschließend möchte ich mich bei allen bedanken, die auf die eine oder andere Art zu diesem Buch beigetragen haben, insbesondere bei meinen Kollegen Konrad Sandau und Udo Häberle von der Hochschule Darmstadt, bei dem Schüler Noah Rabe, bei meiner Ehefrau Renate Ohser-Wiedemann und nicht zuletzt bei Franziska Kaufmann, Christina Kubiak, Manuel Leppert und Frank Katzenmayer vom Carl Hanser Verlag für die sorgfältige redaktionelle Bearbeitung des Manuskripts. Mein Dank gilt auch Lars Thieme, Rainer Günzler, Dieter Horn und Uli Sonntag, mit denen ich in den 1980er Jahren die ersten Schritte in der Bildverarbeitung gemacht habe und von denen ich viele Anregungen zu diesem Buch erhalten habe, vor allem zum letzten Kapitel.

Darmstadt, im Februar 2024 Joachim Ohser

Für

Freja, Carolin, Calla,
Niclas, Hendrik, Oskar, Luis, Per und Mattis

Inhalt

1 Gitter, Bilder und Nachbarschaften

Im folgenden Kapitel sollen einige Grundlagen der Bildverarbeitung und Bildanalyse behandelt werden, die für die Behandlung in den darauf folgenden Kapiteln wichtig sind. Dazu gehört die Einführung digitaler Bilder als ein Sampling von Funktionen auf Gittern. Für viele Algorithmen der Bildverarbeitung und Bildanalyse von Binärbildern werden zudem Nachbarschaften der Pixel vorausgesetzt, die ebenfalls in diesem Kapitel erklärt werden sollen.

1.1 Vorbemerkungen zur Bilddatenstruktur

Um künftig etwas Quellcode zur Notation von Algorithmen in der Programmiersprache `C` präsentieren zu können, verständigen wir uns darauf, dass ein Bild `img` eine Struktur `IMG` mit der inhaltlichen Beschreibung `*description`, der Pixelzahl `*n`, der Pixelgröße `*a`, dem Datentyp `t` der Pixelwerte und den Pixelwerten `**pix` hat, siehe `Image.h`. Dabei ist `**pix` eine rechteckige Matrix von Pixelwerten mit n_1 = `img.n[0]` Zeilen und n_2 = `img.n[1]` Spalten, wobei die Allokation des Speicherplatzes in geeigneter Weise vorzunehmen ist (`Malloc()`) mit den Makros `Malloc.h`. Die Pixelgrößen a_1 = `img.a[0]` und a_2 = `img.a[1]` entsprechen den Gitterabständen rechteckiger Gitter. Ihre Integration in die Bildstruktur ist sinnvoll, da sich die Pixelgröße durch einige Bildverarbeitungsschritte ändert (z. B. bei der Fourier-Transformation) und durch Bildanalyse in der Regel skalierte Kenngrößen (z. B. die Fläche eines Objekts) erhalten werden. Der Daten der Pixelwerte können für `img.t` = 1,2,... vom Typ `unsigned char`, `unsigned short`, `unsigned long`, `float`, `double` etc. seien. Die Pixelwerte von Binärbildern (`img.t` = 0) seien ebenfalls vom Typ `unsigned char`, wobei für Vordergrundpixel lediglich das erste Bit belegt ist. Für Pixel mehrkanaliger Bilder und komplexwertiger Pixel werden später an den entsprechenden Stellen gesonderte Vereinbarungen getroffen werden.

Für eine übersichtlichere Gestaltung von Quelltexten wurde die Struktur `IMG` noch durch die Gesamtzahl `npix` der Pixel des Bildes, der Pixelfläche `apix` und dem Zeiger `str` auf den Datenstream ergänzt; diese Ergänzungen sind aber redundant. Sind also die Pixelwerte `**pix` z. B. vom Typ `unsigned char` und setzen wir einen Datenstream `*str`, der ebenfalls vom Typ `unsigned char` sein muss, mit `str = &((unsigned char **)img->pix)[0][0]` auf die Adresse des ersten Pixels, dann sind die Pixelwerte `pix[i][j]` und `*(str + i * n[0] + j)` für alle Indizes identisch. Formal schreiben wir

```
pix[i][j] ≡ *(str + i * n[0] + j)
```
(1.1)

für `i = 0..n[0]-1` und `j = 0..n[1]-1`, was bedeutet, dass die Pixelwerte auf dem Datenstream y-lokal abgelegt sind. Damit könnte die Invertierung eines 8-Bit-Grautonbildes beispielsweise mit einer Doppelschleife (`InvertImg0()`) oder kürzer und übersichtlicher mit einer

Einfachschleife (`InvertImg()`) geschrieben werden. Wir werden später von beiden Varianten der Pixeladressierung Gebrauch machen. Zudem wird in diesem Buch noch eine Funktion `NewImg()` zur Erzeugung eines Bildes mit einer vorgegebenen Pixelzahl, einer Pixelgröße und eines Datentyps für die Pixelwerte verwendet.

Natürlich müsste in allen Funktionen noch eine sorgfältige Fehlerbehandlung vorgenommen werden, auf die wir hier aber aus Platz- und Übersichtsgründen verzichten wollen.

1.2 Euler-Zahl

Die Euler-Zahl (Euler-Poincaré-Charakteristik) ist zunächst ein Merkmal (*feature*) eines Binärbildes, das algorithmisch sehr einfach bestimmt werden kann und mit dem sich eine Reihe von Problemen der Bildanalyse lösen lässt. In diesem Abschnitt wird die Euler-Zahl jedoch primär dazu eingeführt, um die Bedeutung von Nachbarschaften der Pixel eines Binärbildes und die Komplementarität von Nachbarschaften zu erklären.

Wir betrachten den kontinuierlichen Fall, in dem der Vordergrund und der Hintergrund eines Binärbildes als Teilmengen des 2-dimensionalen Euklidischen Raumes $\mathbb{R}^2$ aufgefasst werden können. Außerdem nehmen wir einfachheitshalber an, dass das Bild unendlich ausgedehnt ist. Das erspart uns die Behandlung von Bildrandeffekten, auf die aber später noch einzugehen ist.

Wenn wir mit $X \subseteq \mathbb{R}^2$ den Vordergrund eines kontinuierlichen Binärbildes bezeichnen, dann ist die Komplementärmenge $X^c = \mathbb{R}^2 \setminus X$ der Hintergrund, wobei $\setminus$ die Mengendifferenz ist, die durch $X \setminus Y = \{x \in \mathbb{R}^2 : x \in X, x \notin Y\}$ für $X, Y \subseteq \mathbb{R}^2$ definiert wird. Die Euler-Zahl wird zunächst für kompakte (d. h. beschränkte und abgeschlossene), konvexe Mengen $K \subset \mathbb{R}^2$ erklärt.

Definition 1.1 Sei $K \subset \mathbb{R}^2$ eine kompakte und konvexe Menge. Die Euler-Zahl $\chi(K)$ der Menge K ist definiert durch

$$\chi(K) = \begin{cases} 1, & \text{falls } K \neq \emptyset \\ 0, & \text{sonst} \end{cases}.$$

Das heißt, $\chi(K)$ nimmt für kompakte, konvexe Mengen nur die Werte 0 oder 1 an. Aus der Einfachheit dieser Definition erklärt sich letztendlich auch die Einfachheit der bildanalytischen Bestimmung. Allerdings bedeutet die Einschränkung auf konvexe Mengen eine wesentliche Einschränkung der Anwendung, denn es kann im Allgemeinen nicht vorausgesetzt werden, dass der Vordergrund eines Binärbildes oder auch nur eine Zusammenhangskomponente des Vordergrundes konvex sind. Wir gehen daher zu nichtkonvexen Mengen über.

1.2.1 Additive Erweiterung

- Die Vereinigung $X = K_1 \cup K_2$ zweier kompakter, konvexer Mengen K_1 und K_2 ist im Allgemeinen nicht konvex. Allerdings ist der Durchschnitt $K_1 \cap K_2$ konvex. Daher kann die Euler-Zahl von X durch

$$\chi(X) = \chi(K_1) + \chi(K_2) - \chi(K_1 \cap K_2)$$

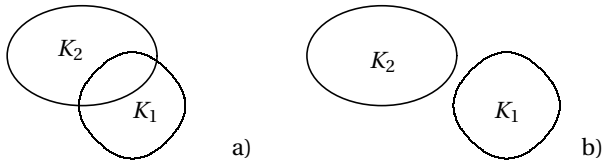


Bild 1.1 Die Vereinigung $X = K_1 \cup K_2$ zweier konvexer Mengen K_1 und K_2 ist im Allgemeinen nicht konvex. a) Der Durchschnitt $K_1 \cap K_2$ ist nicht leer, und folglich ist $\chi(X) = 1$. b) Der Durchschnitt ist leer, und damit ist $\chi(X) = 2$.

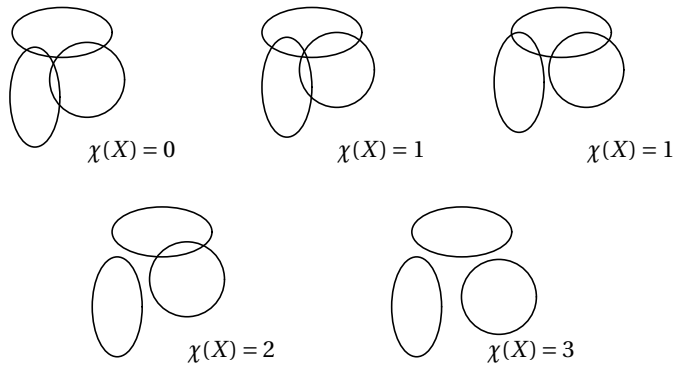


Bild 1.2 Werte für die Euler-Zahl der Vereinigung $X = K_1 \cup K_2 \cup K_3$ dreier konvexer Mengen K_1, K_2 und K_3

erklärt werden. Sind K_1 und K_2 nicht leer, aber disjunkt, dann ist $\chi(X) = 2$. Ist der Durchschnitt $K_1 \cap K_2$ ebenfalls nicht leer (d. h. bildet $K_1 \cup K_2$ eine Zusammenhangskomponente), dann ist $\chi(X) = 1$.

In Bild 1.1 sind die Mengen K_1 und K_2 schematisch dargestellt.

- Für die Vereinigung $X = K_1 \cup K_2 \cup K_3$ dreier Mengen gilt

$$\begin{aligned}\chi(X) \quad = \quad & \chi(K_1) + \chi(K_2) + \chi(K_3) \\ & -\chi(K_1 \cap K_2) - \chi(K_1 \cap K_3) - \chi(K_2 \cap K_3) \\ & +\chi(K_1 \cap K_2 \cap K_3).\end{aligned}$$

Ist X die Vereinigung dreier konvexer Mengen, dann kann die Euler-Zahl von X Werte zwischen 0 und 3 annehmen, siehe Bild 1.2.

- Das lässt sich verallgemeinern. Für eine endliche Vereinigung $X = \bigcup_{i=1}^{m} K_i$ gilt

$$\chi(X) = \sum_{i=1}^{m} \chi(K_i) - \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \chi(K_i \cap K_j) + \ldots - (-1)^m \chi\Big(\bigcap_{i=1}^{m} K_i\Big) \tag{1.2}$$

(Inklusions-Exklusions-Prinzip), wobei anzumerken ist, dass sich jede kompakte, nichtkonvexe Menge hinreichend gut durch eine endliche Vereinigung kompakter konvexer Mengen approximieren lässt (wobei wir hier offen lassen, was „approximieren“ bedeutet).

Wegen der Additivität der Euler-Zahl und des sich daraus ergebenden Inklusions-Exklusions-Prinzips lässt sich die Euler-Zahl lokaler Bildinformation bestimmen.

Die Euler-Zahl ist (im 2-dimensionalen Fall) die Anzahl der Zusammenhangskomponenten minus die Anzahl der „Löcher“.

Aufgabe 1.1 Welche Werte kann die Euler-Zahl der Vereinigung von vier konvexen Mengen annehmen? ■

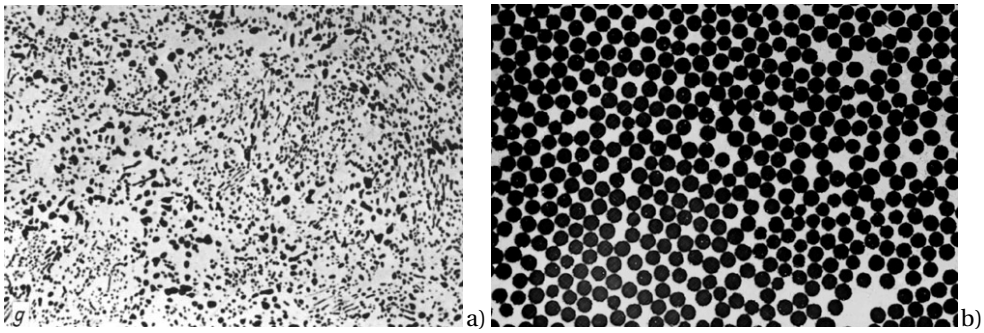


Bild 1.3 a) Zementitausscheidungen (dunkel) in einem eutektoidischen Kohlenstoffstahl in einem ebenen Anschliff, auflichtoptische Aufnahme, 664×948 Pixel, Pixelgröße $a = 0,196\,\mu\text{m}$; b) Verbundwerkstoff, Kohlenstofffasern (dunkel) in Epoxidharz im Querschliff, auflichtoptische Aufnahme, 664×948 Pixel der Größe $a = 0,139\,\mu\text{m}$

Beispiel 1.1 Einfache Zusammenhangskomponenten (einfach zusammenhängende Objekte, Objekte ohne Loch) haben die Euler-Zahl 1. Folglich kann die Euler-Zahl verwendet werden, um einfache Zusammenhangskomponenten zu zählen. Die Zementitausscheidungen in Bild 1.3a sind einfach zusammenhängend. Also entspricht in diesem Fall die gemessene Euler-Zahl χ der Anzahl N der Ausscheidungen: $\chi = 2\,068$ für die 4er-Nachbarschaft, $\chi = 2\,002$ für die 6er-Nachbarschaft und $\chi = 1\,935$ für die 8er-Nachbarschaft (bei einer Binarisierungsschwelle von 128). Meist wird die Anzahl auf die Größe des Bildausschnittes bezogen, d. h., es wird die Anzahl pro Flächeneinheit angegeben, $N_A = 80\,000\,\text{mm}^{-2}$ für die 8er-Nachbarschaft. ■

Beispiel 1.2 Klar, die Anzahl m der (sich berührenden) Kohlenstofffaserquerschnitte in Bild 1.3b kann bestimmt werden, indem die Objekte in einem binarisierten Bild zunächst durch eine morphologische Transformation getrennt und danach gezählt werden, wobei für die Zählung wie oben die Euler-Zahl verwendet werden kann. ■

Beispiel 1.3 Schwieriger scheint es zu sein, in Bild 1.3b die Anzahl n der Faserkontakte (d. h. der Berührungsstellen der Objekte) zu bestimmen. Wir verwenden die Inklusions-Exklusions-Formel (1.2), wobei zu berücksichtigen ist, dass ausschließlich paarweise Überlappungen auftreten. Die Gl. (1.2) kann also vereinfacht werden,

$$\chi(X) = \underbrace{\sum_{i=1}^{m} \chi(K_i)}_{m} - \underbrace{\sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \chi(K_i \cap K_j)}_{n},$$

wobei man sich leicht überlegen kann, dass die Doppelsumme auf der rechten Seite gerade der Anzahl n der Faserkontakte entspricht. Ein Wert für n kann bestimmt werden, wenn neben der Anzahl m der Faserquerschnitte auch die Euler-Zahl der (nicht erodierten) Struktur gemessen und die obige Gleichung nach n umgestellt wird,

$$n = m - \chi(X).$$

Die mittlere Anzahl κ der Faserkontakte pro Faser (d. h. die Koordinationszahl) erhält man aus

$$\kappa = 2\Big(1 - \frac{\chi(X)}{m}\Big).$$

Der Faktor 2 ist dadurch motiviert, dass sich jeweils zwei einander berührende Objekte eine Berührungsstelle „teilen“. Für Bild 1.3b erhält man $\chi = -227$ (bezüglich der 6.1er-Nachbarschaft), $m = 471$ und damit $\kappa = 1{,}036$. ■

Bemerkung 1.1 Ist X eine endliche Vereinigung konvexer Mengen, dann gilt das nicht für ihre Komplementärmenge X^c. Folglich lässt sich die Euler-Zahl von X^c nicht mithilfe des Inklusions-Exklusions-Prinzips erklären. In diesem Fall kann $\chi(X^c)$ durch Hadwigers rekursive Formel definiert werden [108], [193], und es gilt

$$\chi(X) = -\chi(X^c),$$

d. h., durch die Invertierung eines Binärbildes ändert sich das Vorzeichen der Euler-Zahl. ■

1.2.2 Euler-Poincaré-Formel

Für (nichtkonvexe) polygonale Mengen, d. h. für endliche Vereinigungen konvexer Polygone, kann die Euler-Zahl mit der Euler-Poincaré-Formel berechnet werden.

Satz 1.1 Sei X ein (nicht notwendig konvexes oder zusammenhängendes) Polygon mit v Ecken (*vertices*), e Kanten (*edges*) und f konvexen Flächen (*faces*). Für die Euler-Zahl $\chi(X)$ des Polygons gilt

$$\chi(X) = v - e + f \tag{1.3}$$

(Euler-Poincaré-Formel). ■

Beispiel 1.4 Für ein konvexes n-Eck X ist $v = e = n$ und $f = 1$. Somit ist $\chi(X) = n - n + 1 = 1$. ■

Beispiel 1.5 Für die polygonale Menge X in Bild 1.4b ist $v = 15$, $e = 20$ und $f = 5$. Damit erhält man $\chi(X) = 15 - 20 + 5 = 0$. Das Ergebnis ist plausibel, denn in Bild 1.4a ist eine Zusammenhangskomponente mit einem „Loch“ abgebildet. ■

Die Euler-Poincaré-Formel wird in diesem Kapitel vor allem für die bildanalytische Bestimmung der Euler-Zahl gebraucht.

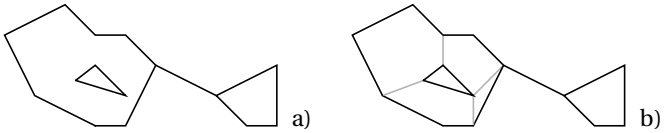


Bild 1.4 a) Eine zusammenhängende polygonale Menge X mit einem „Loch“: Die Euler-Zahl von X ist erwartungsgemäß gleich 0. b) Durch die eingefügten Kanten erhält man eine Unterteilung in konvexe Polygone, deren Vereinigung X ist.

1.2.3 Netzwerkformel

Für viele Anwendungen ist die sogenannte Netzwerkformel hilfreich, die etwas über den Zusammenhang zwischen der Euler-Zahl pro Flächeneinheit χ_A und der Anzahl der Knoten pro Flächeneinheit N_A in einem Netzwerk (d. h. in einem Graphen) aussagt. Wir bezeichnen mit $\bar{\mu}$ die mittlere Anzahl der Kanten des Netzwerks, die von einem Knoten ausgehen. Mit $\bar{\mu}$ wird die mittlere Ordnung der Knoten bezeichnet, also die mittlere Anzahl der Kanten, die von einem Knoten ausgehen. Dann gilt

$$\chi_A = N_A\left(1 - \frac{\bar{\mu}}{2}\right) \tag{1.4}$$

(Netzwerkformel) [178].

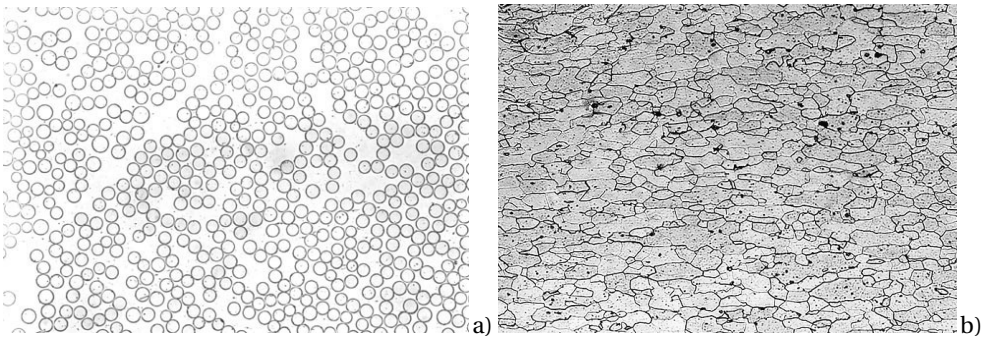


Bild 1.5 a) Glasfasern in Epoxidharz, Querschliff, auflichtmikroskopische Aufnahme, Dunkelfeld, 664×948 Pixel der Größe $a = 0{,}139\,\mu\text{m}$ und b) ferritischer Stahl, Längsschliff eines Drahts, auflichtmikroskopische Aufnahme, Hellfeld, 768×1024 Pixel der Größe $a = 0{,}451\,\mu\text{m}$. Die sich berührenden Kreislinien der Ränder der Glasfasern bzw. das Kantensystem der Ferritkörner bilden Netzwerke.

Beispiel 1.6 Wir fassen die Berührungsstellen der Glasfasern in Bild 1.5a als Knoten eines Netzwerkes auf. Von jedem dieser Knoten gehen vier Kanten aus, und folglich ist $\bar{\mu} = 4$. Vorausgesetzt, das Netzwerk lässt sich segmentieren, dann wird die Euler-Zahl pro Flächeneinheit χ_A bestimmt und die Netzwerkformel (1.4) nach der Anzahl der Berührungsstellen pro Flächeneinheit N_A umgestellt. Man erhält unmittelbar

$$N_A = -\chi_A.$$

■

Beispiel 1.7 Für das Kantensystem in Bild 1.5b ist $\bar{\mu} = 3$. Die Anzahl der Knoten (Kornzwickel) je Flächeneinheit N_V lässt sich folglich mithilfe der Netzwerkformel

$$N_A = -2\chi_A \tag{1.5}$$

berechnen, sofern sich das Kantensystem vernünftig segmentieren lässt und χ_A mit der erforderlichen Genauigkeit bestimmt werden kann. ■

1.3 Homogene Gitter, Sampling und Digitalisierung

Digitale Bilder sind Daten auf Gitterpunkten (*points*, pt), wobei in der Sprache der Bildverarbeitung die Daten den Pixelwerten und die Gitterpunkte den Pixelpositionen entsprechen. Die Menge der Gitterpositionen eines Bildes bildet ein 2-dimensionales Gitter $\mathbb{L}$. Eine besondere Rolle spielen in der Bildverarbeitung homogene Gitter.

Definition 1.2 Ein Gitter $\mathbb{L}$ heißt homogen, wenn es invariant bezüglich Gitterverschiebungen ist, d. h. $\mathbb{L} + x = \mathbb{L}$ für alle $x \in \mathbb{L}$. ■

Seien $u_1, u_2 \in \mathbb{R}^2$ zwei linear unabhängige Vektoren und $\mathbb{Z}$ die Menge der ganzen Zahlen. Dann bildet die Menge

$$\mathbb{L} = \{x \in \mathbb{R}^2 : x = i\,u_1 + j\,u_2,\ i, j \in \mathbb{Z}\}$$

der ganzzahligen Linearkombinationen der Basisvektoren u_1 und u_2 ein homogenes Gitter mit der Einheitszelle $C = \{x \in \mathbb{R}^2 : x = p\,u_1 + q\,u_2,\ 0 \le p, q \le 1\}$, d. h., C hat die Eckpunkte 0, u_1, u_2 und $u_1 + u_2$. Die Kantenlängen $a_1 = \|u_1\|$ und $a_2 = \|u_2\|$ sind die Gitterabstände, die den Pixelgrößen entsprechen. Fassen wir die beiden Basisvektoren u_1 und u_2 zu einer Matrix $U = (u_1, u_2)$ zusammen, dann ist $\text{Fläche}(C) = |\det U|$ die Fläche der Einheitszelle C, die häufig mit der Pixelgröße eines digitalen Bildes assoziiert wird. Homogene Gitter sind unbeschränkt; in Bild 1.6a wird lediglich ein Ausschnitt gezeigt.

Das Gitter $\mathbb{L}$ ist durch seine Basisvektoren charakterisiert, jedoch sind für ein gegebenes homogenes Gitter die Basisvektoren nicht eindeutig bestimmt. Für ein inhomogenes Gitter, siehe Bild 1.6b, gibt es keine Basis. Ein inhomogenes Gitter ist durch eine Basis nicht charakterisierbar, weshalb solche Gitter für die Bildverarbeitung ungeeignet sind.

Beispiel 1.8 Quadratische Gitter $\mathbb{L} = a\mathbb{Z}^2$ mit einem Gitterabstand $a > 0$ werden z. B. in CCD- und CMOS-Kameras zugrunde gelegt, Bild 1.7a. Die gebräuchliche Gitterbasis ist

$$u_1 = \begin{pmatrix} a \\ 0 \end{pmatrix}, \qquad u_2 = \begin{pmatrix} 0 \\ a \end{pmatrix}.$$

Alternativ könnte aber auch die Basis

$$u_1 = \begin{pmatrix} a \\ 0 \end{pmatrix}, \qquad u_2 = \begin{pmatrix} a \\ a \end{pmatrix}$$

gewählt werden. ■

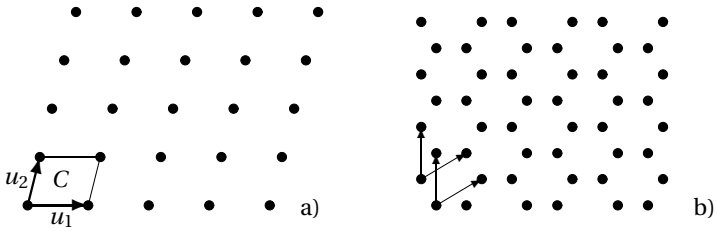


Bild 1.6 a) Ausschnitt aus einem homogenen 2-dimensionalen Gitter $\mathbb{L}$ mit 25 Punkten (25 pt), den Basisvektoren u_1 und u_2 und der Einheitszelle C; b) Beispiel für ein inhomogenes Gitter als Superposition zweier homogener Gitter

Aufgabe 1.2 Häufig wird die laterale Auflösung in der Maßeinheit Pixel pro inch (*dots per inch*, dpi) angegeben. Welchen Wert hat die Pixelgröße a eines quadratischen Gitters bei einer lateralen Auflösung von 600 dpi? ■

Beispiel 1.9 Zeilenkameras liefern meist Bilder auf rechteckigen Gittern, die z. B. durch die Basisvektoren

$$u_1 = \begin{pmatrix} a_1 \\ 0 \end{pmatrix}, \qquad u_2 = \begin{pmatrix} 0 \\ a_2 \end{pmatrix}$$

charakterisiert sind. In Bild 1.7b ist das Verhältnis der Seitenlängen $a_1/a_2 = 1{,}5$. Die Pixelgröße a_1 entspricht der lateralen Auflösung in Zeilenrichtung, und a_2 ist von der Relativbewegung abzubildener Objekte orthogonal zur Zeilenrichtung abhängig (also z. B. von der Geschwindigkeit eines Förderbandes, über dem die Zeilenkamera angebracht ist. ■

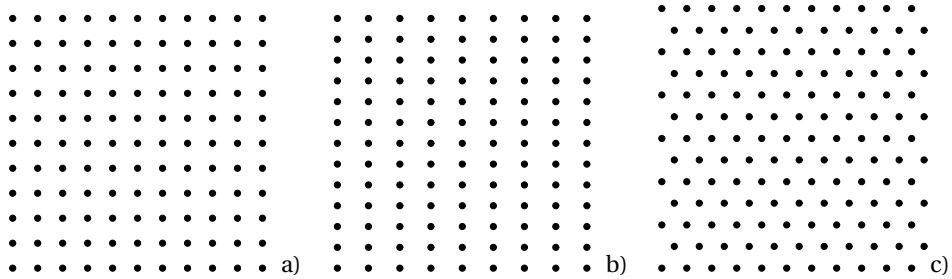


Bild 1.7 Ausschnitte homogener Gitter: a) quadratisch, b) rechteckig, c) hexagonal

Beispiel 1.10 Ein Gitter mit der Basis

$$u_1 = \begin{pmatrix} a \\ 0 \end{pmatrix}, \qquad u_2 = \frac{1}{2}\begin{pmatrix} a \\ \sqrt{3}a \end{pmatrix}$$

und dem einheitlichen Gitterabstand $a > 0$ ist hexagonal, siehe Bilder 1.7c und 1.8a. In den 1970er Jahren wurden von der Firma Leitz Kameras mit hexagonalen Pixelrastern eingeführt, was für die Bildverarbeitung und Bildanalyse Vorteile im Vergleich zu

quadratischen Rastern hat. Die Grundlagen dafür gehen auf Entwicklungen des Centre de Morphologie Mathématique in Fontainebleau zurück, siehe [250] und darin zitierte Literatur. ■

Bei der Fourier-Transformation von Bildern spielt das inverse Gitter eine Rolle, das im Zusammenhang mit der Beugung von Röntgenstrahlen an Kristallgittern auch als reziprokes Gitter bezeichnet wird. Die Matrix $\hat{U}$ der Gitterbasis des zu $\mathbb{L}$ inversen Gitters $\hat{\mathbb{L}}$ ist die Transponierte der Inversen von U,

$$\hat{U} = (U^{-1})',$$

wobei die Spaltenvektoren $\hat{u}_1$ und $\hat{u}_2$ von $\hat{U}$ die Basis des inversen Gitters bilden, siehe auch Abschnitt 4.4.3. Offensichtlich haben die Gitterabstände $\|\hat{u}_1\|$ und $\|\hat{u}_2\|$ von $\hat{\mathbb{L}}$ die Maßeinheit m^{-1}, wenn die Maßeinheit der Gitterabstände $\|\hat{u}_1\|$ und $\|\hat{u}_2\|$ in m gegeben ist. Für die Pixelgröße des inversen Gitters mit der Einheitszelle $\hat{C}$ gilt $F(\hat{C}) = 1/F(C)$.

Beispiel 1.11 Das zum hexagonalen Gitter $\mathbb{L}$ inverse Gitter $\hat{\mathbb{L}}$ ist ebenfalls hexagonal. Aus der in Beispiel 1.10 gegebenen Basis folgt

$$U = \frac{a}{2}\begin{pmatrix} 2 & 1 \\ 0 & \sqrt{3} \end{pmatrix}, \qquad U^{-1} = \frac{1}{\sqrt{3}a}\begin{pmatrix} \sqrt{3} & -1 \\ 0 & 2 \end{pmatrix}, \qquad \hat{U} = \frac{1}{\sqrt{3}a}\begin{pmatrix} \sqrt{3} & 0 \\ -1 & 2 \end{pmatrix}.$$

Die Basisvektoren $\hat{u}_1$ und $\hat{u}_1$ des inversen Gitters sind also

$$\hat{u}_1 = \frac{1}{\sqrt{3}a}\begin{pmatrix} \sqrt{3} \\ -1 \end{pmatrix}, \qquad \hat{u}_2 = \frac{1}{\sqrt{3}a}\begin{pmatrix} 0 \\ 2 \end{pmatrix}.$$

In Bild 1.8 sind die beiden zueinander inversen Gitter $\mathbb{L}$ und $\hat{\mathbb{L}}$ dargestellt. Die Pixelgröße von $\mathbb{L}$ ist $a_1 = a_2 = a$, die des inversen Gitters ist $a_1 = a_2 = \frac{2}{\sqrt{2}a}$. ■

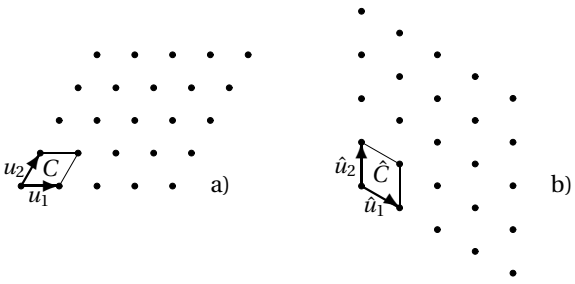


Bild 1.8 Ausschnitte a) aus dem hexagonalen Gitter $\mathbb{L}$ und b) dem zugehörigen inversen Gitter $\hat{\mathbb{L}}$

Aufgabe 1.3 Gegeben sei ein Gitter $\mathbb{L}$ mit den Basisvektoren

$$u_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad u_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Bestimmen Sie eine Basis des zu $\mathbb{L}$ inversen Gitters $\hat{\mathbb{L}}$. ■

Aufgabe 1.4 Unter welcher Voraussetzung ist $\mathbb{L} = \hat{\mathbb{L}}$? ■

Als einfachstes Modell der Abtastung wird das Sampling $X_\square = X \cap \mathbb{L}$ einer Menge $X \subset \mathbb{R}^2$ auf einem homogenen Gitter $\mathbb{L}$ eingeführt. Es besteht aus allen Gitterpunkten, die in X liegen, Bild 1.9b. Unter $X \cap \mathbb{L}$ kann die Menge der Vordergrundpixel eines digitalen Binärbildes verstanden werden, und das Sampling $X^c \cap \mathbb{L}$ der Komplementärmenge X^c ist die Menge der Hintergrundpixel, Bild 1.9c. Sampling ist in der Regel mit einem Informationsverlust verbunden. Es ist also im Allgemeinen nicht möglich, X aus dem Sampling $X \cap \mathbb{L}$ zu rekonstruieren, siehe Abschnitt 4.3.2.

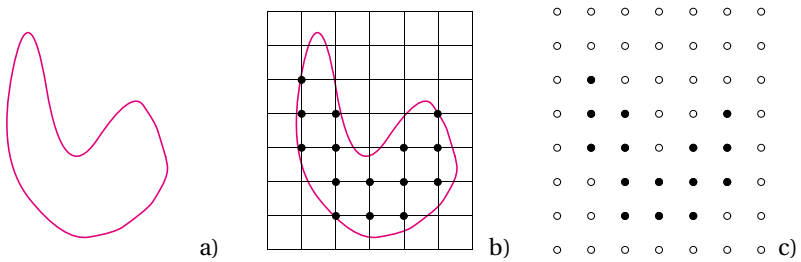


Bild 1.9 a) Eine Menge X, b) ihr Sampling $X \cap \mathbb{L}$ auf einem quadratischen Gitter $\mathbb{L}$ und c) das zugehörige Binärbild, wobei • die Vorder- und ∘ die Hintergrundpixel kennzeichnen

Digitalisierungen sind Approximationen von X, die aus einer Abtastung $X \cap \mathbb{L}$ von X auf dem Gitter $\mathbb{L}$ erzeugt werden.

- Die Gauß-Digitalisierung ist die Vereinigung aller Gitterzellen, deren untere linke Ecke in X liegt,

$$\bigcup_{x \in X \cap \mathbb{L}} (C + x), \tag{1.6}$$

  Bild 1.10a.

- Die innere Jordan-Digitalisierung ist die Vereinigung aller Gitterzellen, die vollständig in X liegen,

$$\bigcup_{x \in \mathbb{L}} \{C + x : X \subseteq C + x\},$$

  Bild 1.10b.

- Die äußere Jordan-Digitalisierung ist die Vereinigung aller Gitterzellen, die von X geschnitten werden,

$$\bigcup_{x \in \mathbb{L}} \{C + x : X \cap (C + x) \neq \emptyset\},$$

  Bild 1.10c.

- Betrachtet wird ein Rendering von X, das als Polygonzug die Mittelpunkte aller Kanten der Gitterzellen von $\mathbb{L}$ verbindet, für die jeweils ein Ende in X und das andere außerhalb von X liegt. Das zugehörige Polygon kann als eine Digitalisierung von X aufgefasst werden, Bild 1.10d.

Während die Gauß-Digitalisierung und das Rendering direkt aus dem Sampling $X \cap \mathbb{L}$ erzeugt werden, können die beiden Jordan-Digitalisierungen im Allgemeinen nicht aus $X \cap \mathbb{L}$ erhalten werden. Näherungen der inneren und äußeren Jordan-Digitalisierung sind die Vereinigung aller Gitterzellen, für die alle Eckpunkte bzw. mindestens ein Eckpunkt in X liegen.

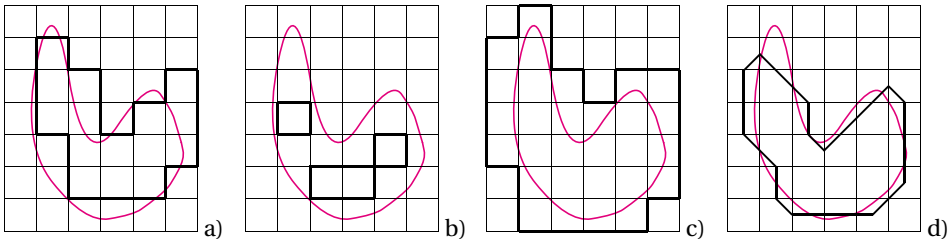


Bild 1.10 Digitalisierungen eines Objekts bezüglich eines quadratischen Gitters $\mathbb{L}$: a) Gauß-Digitalisierung, b) innere Jordan-Digitalisierung, c) äußere Jordan-Digitalisierung, d) das Polygon, dessen Rand durch Rendering von X auf $\mathbb{L}$ erhalten wurde

Bemerkung 1.2 Rendering spielt in der 3D-Bildverarbeitung eine große Rolle bei der Visualisierung von 3D-Binärbildern. Das in Bild 1.10 gezeigte Rendering des Vordergrundes eines 2D-Binärbildes entspricht dem Oberflächenrendering (indirektes Rendering) von 3D-Binärbildern [194]. ■

1.4 Lokale Pixelkonfigurationen

In diesem Abschnitt werden lokale Pixelkonfigurationen von Binärbildern eingeführt [105, 106, 107, 101, 250] Wie wir später noch sehen werden, enthält die Anzahl dieser Konfigurationen wesentliche Informationen. Wichtig sind vor allem 2×2-Pixelkonfigurationen (*bit-quads*), aus deren Anzahlen unter anderem die Euler-Zahl eines Binärbildes bestimmt werden kann.

Man kann sich leicht überlegen, dass in einem Binärbild 16 voneinander verschiedene 2×2-Pixelkonfigurationen vorkommen können. Zur Bezeichnung dieser Pixelkonfigurationen werden Piktogramme eingeführt, die in Tabelle 1.1 zusammengefasst sind.

Um jeder 2×2-Pixelkonfiguration einen Index k zuzuordnen, wird wie folgt verfahren: Sei $B = (b_{ij})$ die Matrix der Pixelwerte b_{ij} eines Binärbildes, wobei die Pixel die Werte 0 für den Hintergrund oder 1 für den Vordergrund haben. Außerdem wird die Filtermaske

$$M = \begin{pmatrix} 2 & 1 \\ 8 & 4 \end{pmatrix}$$

eingeführt, deren Koeffizienten Zweierpotenzen sind. Das farbig markierte Pixel (Offsetpixel) liegt im Koordinatenursprung. Die lineare Filterung $B * M$ des Binärbildes B mit der Maske M ergibt ein 4-Bit-Grautonbild $F = (f_{ij})$. Hierbei ist zu beachten, dass die Filterung (Faltung) einer Korrelation mit der gespiegelten Maske

$$M^* = \begin{pmatrix} 4 & 8 \\ 1 & 2 \end{pmatrix}$$

Tabelle 1.1 Die 2×2-Pixelkonfigurationen von Binärbildern und die zugehörigen Koeffizienten w_k zur Berechnung der Euler-Zahl in Abhängigkeit von der Wahl der Nachbarschaft: Die Tabelle enthält zur besseren Übersicht das Vierfache $4w_k$ der Koeffizienten.

		$4w_k$		
k	Pikt.	4er	6er	8er
0		0	0	0
1		1	1	1
2		1	1	1
3		0	0	0
4		1	1	1
5		0	0	0
6		2	-2	-2
7		-1	-1	-1

		$4w_k$		
k	Pikt.	4er	6er	8er
8		1	1	1
9		2	2	-2
10		0	0	0
11		-1	-1	-1
12		0	0	0
13		-1	-1	-1
14		-1	-1	-1
15		0	0	0

entspricht, $B * M = B \star M$, siehe Abschnitt 2.2. Aufgrund der Wahl der Koeffizienten von M wird jeder Pixelkonfiguration in B in eineindeutiger Weise ein Grauwert f_{ij} zugeordnet, und wir legen fest, dass dieser Grauwert der Index k der Pixelkonfiguration ist, die sich an der Stelle (i, j) von B befindet. Die Pixelkonfiguration  hat folglich den Index $k = 13$.

Beispiel 1.12 Setzt man ein Padding des Binärbildes mit Nullen voraus, dann erhält man aus

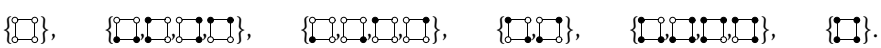
$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

das Grautonbild

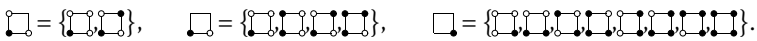
$$F = B * M = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 3 & 1 & 0 \\ a & d & e & 5 & 0 \\ a & 7 & b & 5 & 0 \\ 8 & c & c & 4 & 0 \end{pmatrix},$$

wobei die Pixelwerte von F für eine komprimierte Schreibweise als Hexadezimalzahlen angegeben sind. ■

Für manche Zwecke ist es sinnvoll, Pixelkonfigurationen zu Kongruenzklassen zusammenzufassen. Jede dieser Klassen enthält alle Konfigurationen, die sich durch Drehungen ineinander überführen lassen. Es gibt sechs Kongruenzklassen:



Außerdem sollen im Folgenden auch Klassen von Konfigurationen durch Piktogramme beschrieben werden, in denen Pixel entweder Vorder- oder Hintergrund sind. Das soll an Beispielen klargemacht werden,



Schließlich wird noch der Vektor $h = (h_k)$ der Anzahl von Konfigurationen im Binärbild B eingeführt, wobei mit h_k die Anzahl der Konfigurationen mit dem Index k bezeichnet wird, $k = 0, \ldots, 15$. Mit h wird das Grauwerthistogramm des Grautonbildes F bezeichnet, das durch eine lineare Filterung des Binärbildes B mit M erhalten wird, siehe Gl. (2.20), d. h., h_k ist die Anzahl der Pixel in F mit dem Grauwert $f_{ij} = k$, siehe auch Gl. (1.14). Mit # wird die Anzahl einer oder mehrer Konfigurationen bezeichnet, womit man z. B.

$$\#(\,\ulcorner\!\urcorner\,) = h_{11}, \qquad \#(\,\ulcorner\,\urcorner\,) = h_4 + h_5 + h_6 + h_7, \qquad \#(\,\ulcorner\,\urcorner\,) = \sum_{k=8}^{15} h_k$$

schreiben kann. Die Summe der h_k ist gleich der Pixelzahl n eines Binärbildes, $n = \#(\square) = \sum_{k=0}^{15} h_k$.

Beispiel 1.13 Für das Binärbild

$$B = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

mit $n = 256$ Pixeln erhält man

$$F = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 2 & 3 & 3 & 3 & 1 & 0 & 2 & 3 & 1 & 2 & 1 & 2 & 1 & 0 \\
a & 5 & 8 & c & c & c & 6 & 1 & a & f & 5 & 8 & 4 & a & 5 & 0 \\
a & 7 & 3 & 1 & 0 & 0 & a & 7 & b & d & 6 & 1 & 2 & b & 5 & 0 \\
a & f & d & 6 & 1 & 0 & a & d & e & 5 & a & 5 & 8 & e & 5 & 0 \\
8 & c & 4 & a & 7 & 3 & 9 & 6 & 9 & 6 & 9 & 6 & 3 & b & 5 & 0 \\
0 & 2 & 1 & a & d & e & 5 & 8 & 6 & 9 & 6 & 9 & c & c & 4 & 0 \\
0 & 8 & 6 & b & 5 & a & 5 & 0 & 8 & 4 & a & 5 & 0 & 0 & 0 & 0 \\
0 & 2 & b & d & 4 & 8 & 6 & 3 & 3 & 1 & a & 5 & 0 & 0 & 0 & 0 \\
2 & 9 & c & 6 & 3 & 3 & 9 & e & d & 6 & b & 7 & 3 & 3 & 1 & 0 \\
a & 7 & 3 & b & f & d & 6 & b & 5 & a & d & e & f & d & 4 & 0 \\
8 & e & d & e & f & 7 & 9 & c & 4 & a & 5 & a & d & 6 & 1 & 0 \\
0 & a & 5 & a & f & f & 7 & 1 & 0 & 8 & 6 & 9 & 6 & 9 & 4 & 0 \\
0 & 8 & 4 & 8 & e & d & e & 7 & 3 & 3 & b & 5 & 8 & 6 & 1 & 0 \\
2 & 1 & 2 & 3 & 9 & 6 & 9 & e & f & f & f & 7 & 1 & 8 & 4 & 0 \\
8 & 4 & 8 & c & 4 & 8 & 4 & 8 & c & c & c & c & 4 & 0 & 0 & 0
\end{pmatrix}, \quad h = \begin{pmatrix}
50 \\ 17 \\ 11 \\ 17 \\ 14 \\ 17 \\ 18 \\ 9 \\ 18 \\ 12 \\ 19 \\ 9 \\ 13 \\ 12 \\ 10 \\ 10
\end{pmatrix},$$

wobei wieder Padding mit Nullen angenommen wurde. ■

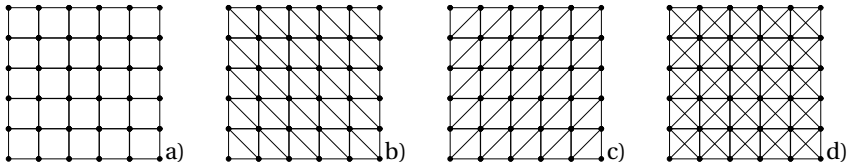


Bild 1.11 Ein Ausschnitt eines quadratisches Gitters mit verschiedenen Nachbarschaftsgraphen: a) 4er-, b) 6.1er-, c) 6.2er- und d) 8er-Nachbarschaft

Natürlich kann der Vektor h auch direkt aus B berechnet werden, ohne F explizit als Zwischenergebnis zu erhalten. In der Funktion `number_of_configs()` ist die Idee eines besonders effektiven Algorithmus implementiert. Dieser Algorithmus hat die Komplexität $\mathcal{O}(n)$, wobei $n = n_1 n_2$ die Pixelzahl ist. Hier wird die Indizierung der 2×2-Konfigurationen temporär durch die Maske

$$M = \begin{pmatrix} 4 & 1 \\ 8 & 2 \end{pmatrix}$$

festgelegt, um den Index k bei der Verschiebung der Maske M durch das Bitshift `k <<= 2` besonders effektiv zu berechnen. Das erfordert jedoch eine nachträgliche Vertauschung einiger Komponenten von h mithilfe des Makros `SWAP`.

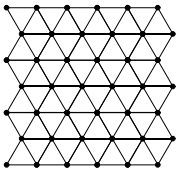
Aufgabe 1.5 Geben Sie den Vektor h für das in Beispiel 1.12 verwendete Binärbild B an. Bestimmen Sie $\#\left(\begin{smallmatrix}\square\\ \bullet\end{smallmatrix}\right)$. ■

Abschließend wird noch darauf hingewiesen, dass der Vektor $h^c = (h_k^c)$ mit den Koeffizienten $h_k^c = h_{15-k}$ der Vektor der Anzahlen der 2×2-Pixelkonfigurationen des invertierten Binärbildes ist.

1.5 Nachbarschaften von Pixeln und ihre Komplementarität

Viele Algorithmen der Bildverarbeitung und Bildanalyse sind von der Wahl der Nachbarschaft der Pixel abhängig. Dazu zählen das Labeling, die Wasserscheidentransformation, die Skelettierung, die Berechnung geodätischer Abstände, die bildanalytische Bestimmung der Euler-Zahl und das Rendering. (Das Rendering in Bild 1.10d bezieht sich auf die 4er-Nachbarschaft.) Da von diesen Algorithmen die Bestimmung der Euler-Zahl sicherlich der einfachste Algorithmus ist, lässt sich deren Implementierung in Abhängigkeit von der Wahl der Nachbarschaft besonders anschaulich darstellen. Außerdem kann die Komplementarität von Nachbarschaften sehr gut mithilfe der Euler-Zahl erklärt werden.

In Bild 1.11 wird eine Übersicht über gebräuchliche Nachbarschaften gegeben, die sich anschaulich durch ihre Nachbarschaftsgraphen darstellen lassen. Da die beiden 6er-Nachbarschaften bis auf eine Drehung äquivalent sind, beschränken wir uns im Folgenden auf die 6.1er-Nachbarschaft; mit 6er-Nachbarschaft ist also stets die 6.1-er Nachbarschaft gemeint.


a)

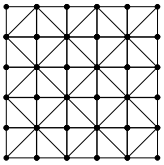
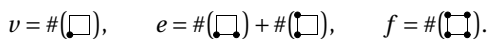

b)

Bild 1.12 a) Die 6er-Nachbarschaft für das hexagonale Gitter, b) die 4-8er-Nachbarschaft, d. h. einer Komnination aus 4er- under 8er-Nachbarschaft

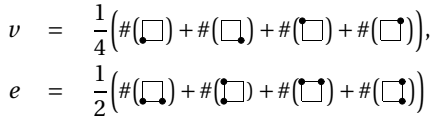
Für das hexagonale Gitter ist die 6er-Nachbarschaft offensichtlich die natürliche Wahl, Bild 1.12a, und schließlich möchten wir noch darauf hinweisen, dass es zulässig ist, die Nachbarschaft wie z. B. Bild 1.12b von Pixel zu Pixel zu wechseln.

Wegen ihrer Additivität kann die Euler-Zahl mithilfe der Euler-Poincaré-Formel (1.3) aus Anzahlen von 2×2-Pixelkonfigurationen bestimmt werden.

Beispiel 1.14 Wir betrachten zunächst die 4er-Nachbarschaft. Die Anzahlen der Ecken, Kanten bzw. Zellen sind



Man kann sich leicht überlegen, dass aus Symmetriegründen die Ecken- und Kantenzahl auch mithilfe der Mittelwerte

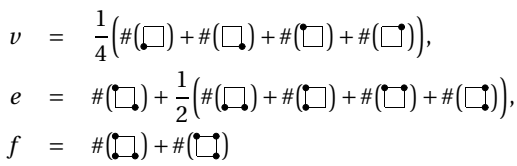


berechnet werden können, was bis auf Bildrandeffekte identisch mit der Verwendung der obigen Gleichungen ist. Aus Gl. (1.3) erhält man die Euler-Zahl χ eines Binärbildes, für dessen Pixel die 4er-Nachbarschaft angenommen wird,

$$\chi = \frac{1}{4}(h_1 + h_2 + h_4 + h_8) + \frac{1}{2}(h_6 + h_9) - \frac{1}{4}(h_7 + h_{11} + h_{13} + h_{14}).$$

Die gewichtete Summe auf der rechten Seite dieser Gleichung kann übersichtlicher mit einem passenden Gewichtsvektor $w = (w_k)$ als Skalarprodukt $\chi = hw$ geschrieben werden, wobei die Gewichte w_k in der Tabelle 1.1 zusammengefasst sind. ■

Beispiel 1.15 Für die 6er-Nachbarschaft führt der Ansatz



zum Ziel. Mit Gl. (1.3) erhält man

$$\chi = \frac{1}{4}(h_1 + h_2 + h_4 + 2h_9 + h_8) - \frac{1}{4}(2h_6 + h_7 + h_{11} + h_{13} + h_{14}).$$

Die entsprechenden Gewichte sind ebenfalls in Tabelle 1.1 enthalten. ■

Offensichtlich wird die Eulerzahl mit zunehmender Nachbarschaft kleiner.

Aufgabe 1.6 Wie werden v, e und f zur Berechnung der Euler-Zahl bezüglich der 8er-Nachbarschaft gewählt? Berechnen Sie die Koeffizienten w_k. ■

Beispiel 1.16 Mit den Koeffizienten aus Tabelle 1.1 erhält man für das Binärbild in Beispiel 1.13 mit $\chi = hw$ die Euler-Zahlen $\chi = 20$, $\chi = 2$ und $\chi = -10$ für die 4er-, 6er- bzw. 8er-Nachbarschaft. ■

Die Koeffizienten w_k sind unabhängig vom Bildinhalt und vom Gitter $\mathbb{L}$. Die Gewichte in Tabelle 1.1 können daher für Binärdatensätze auf beliebigen homogenen Gittern verwendet werden. Die Funktion `EulerNumber()` ist eine geeignete Implementierung, die auf `number_of_configs()` basiert.

Das letzte Beispiel zeigt, dass die gemessene Euler-Zahl stark von der Wahl der Nachbarschaft abhängig sein kann. Das trifft insbesondere auf Bilder mit geringer lateraler Auflösung zu. Die Nachbarschaft muss in diesen Fällen also sorgfältig gewählt werden. Mit zunehmender Nachbarschaft wird die Euler-Zahl kleiner. Darüber hinaus sollte zur Kenntnis genommen werden, dass dann, wenn eine Nachbarschaft für die Vordergrundpixel gewählt wurde, implizit auch eine Nachbarschaft für die Hintergrundpixel festgelegt ist. Die beiden Nachbarschaften für den Vorder- und Hintergrund bilden ein Paar komplementärer Nachbarschaften, Bild 1.13. Aber welche Nachbarschaften sind komplementär zueinander?

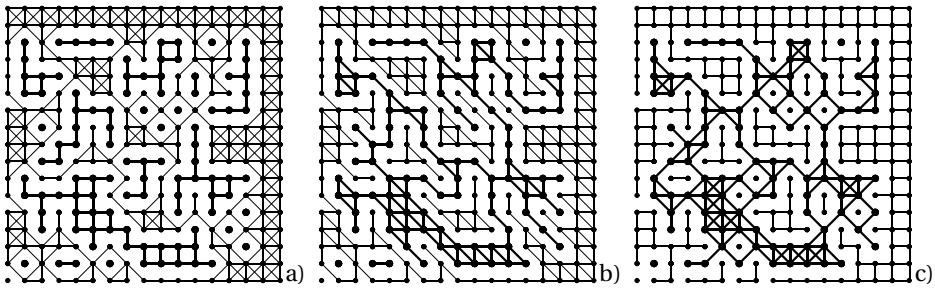


Bild 1.13 Komplementäre Nachbarschaftsgraphen des Binärbildes B aus Beispiel 1.13: a) 4er-, b) 6er- und c) 8er-Nachbarschaft. Eingezeichnet sind auch die Nachbarschaftsgraphen der Hintergrundpixel für die komplementären Nachbarschaften. Beide Nachbarschaftsgraphen – der des Vorder- und der des Hintergrundes – schneiden einander nicht.

Mit w und w^c werden die Gewichtsvektoren zur Berechnung der Euler-Zahl bezüglich eines Paares komplementärer Nachbarschaften bezeichnet. Da die Euler-Zahlen von Vorder- und Hintergrund sich nur durch das Vorzeichen unterscheiden, muss gelten

$$\chi = hw = -h^c w^c,$$

wobei $h^c = (h_k^c)$ der Vektor der Anzahl der 2×2-Konfigurationen des invertierten Binärbilds ist. Aus $h_k^c = h_{15-k}$ folgt unmittelbar $w_k^c = -w_{15-k}$. Wir fassen diese Aussage zu einem Satz zusammen: