Technische Universität Wien

DIPLOMARBEIT

Iterierte Funktionensysteme – Das eindimensionale inverse Problem

ausgeführt am Institut für

Automation

Abteilung für

Mustererkennung und Bildverarbeitung

der Technischen Universität Wien

unter Anleitung von O.Univ.Prof. Dr. W. KROPATSCH

durch

Irene J. Leitgeb

Matrikelnummer 8625086

Heiligenstädter Straße 133/3/37, A–1190 Wien

21. Mai 1993 .....

Zusammenfassung

Da Bilder natürlicher Objekte oft fraktale Züge haben, scheinen Iterierte Funktionensysteme (IFS) zu ihrer Darstellung sehr geeignet. Ein weiterer Grund für die Verwendung von IFS ist deren geringer Speicherbedarf. Ein IFS ist eine Menge von kontraktiven Funktionen, die einen fraktalen Attraktor definiert; dieser Attraktor kann als binäres Bild interpretiert werden. In dieser Arbeit wird zunächst eine anschauliche Erklärung von IFS und dem inversen Problem gegeben. Es folgt ein Überblick über bestehende Ansätze zur Lösung des inversen Problems und eine kurze Einführung in die mathematischen Grundlagen von IFS. Dann wird ein Algorithmus zur Lösung des inversen Problems im eindimensionalen Fall vorgestellt, der auf einer bereits existierenden Diskretisierung der Transformationen eines IFS und Algorithmen zur Berechnung eines diskreten Attraktors basiert. Dabei wird das Verhältnis der Länge von schwarzen und weißen Zusammenhangskomponenten eines Attraktors verwendet, das in $\mathbb{R}$ unter den Transformationen eines IFS invariant ist. Im diskreten Fall kann ein Intervall für dieses Verhältnis definiert werden. Diese Information dient dazu, mögliche Transformationen zu bestimmen. Unter Berücksichtigung der Effekte beim Abtasten findet der Algorithmus dann für jede mögliche Transformation die Parameter, falls diese existieren.

Danksagung

Ich danke Prof. Dr. Kropatsch herzlich für seine Unterstützung und seine Betreuung dieser Arbeit. Ebenso danke ich Dr. Pinz und Dipl. Ing. Bischof sowie den anderen Mitarbeitern der Abteilung.

Abstract

Since images of natural objects often have fractal features, Iterated Function Systems (IFS) seem quite suited for their representation in computers. A further reason to use IFS is their low requirements of memory. An IFS is a set of contractive functions that defines a fractal attractor; this attractor can be interpreted as a binary image. First of all, an explanation of IFS and the inverse problem will be given in this work, followed by an outline of existing approaches to the solution of the inverse problem. A brief introduction to the mathematical basis of IFS will then be given. Then an algorithm is presented that is based upon existing discretisation of transformations of an IFS and algorithms that find a discrete attractor of an IFS. The algorithm solves the inverse problem in the 1D case. It uses the ratio of the length of the black and the white connected components of an attractor which is invariant under the transformations of an IFS in $\mathbb{R}$. In the discrete case an interval can be defined for this ratio. This information is used to find possible transformations. In consideration of sampling effects, the algorithm tries then to find the parameters for each possible transformation if they exist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

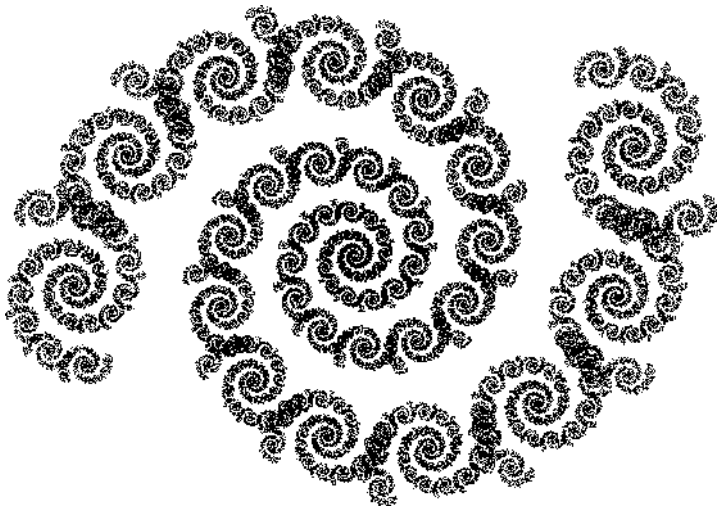


Abbildung 1.1: Ein fraktales Bild

Das Bild in Abb. 1.1 ist ein fraktales Bild. Die herausragende Eigenschaft eines solchen Bildes ist seine Selbstähnlichkeit: In einem fraktalen Bild treten gleiche oder ähnliche, aber jedenfalls verkleinerte Kopien seiner selbst auf. In Abb. 1.1 ist diese Tatsache deutlich zu sehen. Weitere typische Eigenschaften sind die Komplexität und der große Detailreichtum dieser Bilder. Genaueres über Fraktale ist in [21] nachzulesen.

Bilder fraktaler Natur treten häufiger auf, als man der Beschreibung nach vermuten könnte. Besonders Abbildungen aus der Natur haben oft fraktale Züge. Man denke sich eine Gebirgslandschaft: ein Gebirge setzt sich aus Bergen zusammen, die sich wiederum aus großen Felsen zusammensetzen, die von kleinen Felsen gebildet werden... Man könnte diese Hierarchie bis hinunter zu kleinen Steinen verfolgen. Auch viele Bilder von Pflanzen, speziell Bäumen, oder von Wolken sind fraktaler Natur.

In unserer Zeit besteht nun die Tendenz, anstelle von Text immer mehr Bilder auf dem Computer

zu zeigen. Man hat erkannt, daß Bilder die Anschaulichkeit und Attraktivität von Programmen erhöhen. Es existiert einerseits der Bereich der Bildverarbeitung, die Information aus Bildern anderer Quellen (z.B. Luftaufnahmen) gewinnt, und andererseits das Gebiet der Computergraphik, die komplexe Bilder am Computer generiert.

In beiden Bereichen steigt die Anzahl der verwendeten fraktalen Bilder; in der Computergraphik werden z.B. Fraktale zur Erzeugung realistischer Bilder verwendet [8, 10, 11]. Dies bringt natürlich einen erhöhten Verbrauch von Speicherplatz mit sich. Deshalb ist es von Interesse, Bilder u.a. mit möglichst hohen Kompressionsraten zu codieren.

Hier soll weder auf Codierung im allgemeinen noch auf Codierungsmethoden im einzelnen eingegangen werden; eine Übersicht findet man in [15]. Wir möchten jedoch fraktale Bilder auf eine Reihe von Bildeigenschaften untersuchen, die i.a. bei der Codierung ausgenutzt werden.

Hier eine kurze Auflistung dieser Bildeigenschaften:

- Möglichst lange Folgen von Pixeln gleicher Farbe in einem Bild, sogenannte Läufe.
- Konturen innerhalb eines Bildes
- Möglichst große Blöcke gleicher Farbe (z.B. Quadtree [6])
- Korrelation der Pixel

Betrachtet man nun das fraktale Bild in Abb. 1.1, wird man feststellen, daß diese Eigenschaften aufgrund der vielen vorhandenen Details nicht sehr ausgeprägt sind. Codierverfahren, die mit solchen Bildeigenschaften arbeiten, bringen also für fraktale Bilder meist keine guten Resultate.

Außer diesen gibt es noch Verfahren, die das zu codierende Bild vom Orts– in den Frequenzbereich transformieren (z.B. Blockquantisierung [9] oder Wavelets [20]). Hier wird die Tatsache ausgenutzt, daß das menschliche Auge auf hohe Frequenzen weniger sensibel reagiert als auf niedrige. Die hohen Frequenzen werden deshalb bei der Codierung vernachlässigt. Da bei einem fraktalen Bild wegen seines Detailreichtums großes Gewicht auf den hohen Frequenzen liegt, ist auch eine Codierung dieser Art nicht ideal.

1.1 Fraktale Codierung

Man sieht also, daß fraktale Bilder durch herkömmliche Codierungsmethoden nur mit relativ niedriger Kompressionsrate codierbar sind. Aus diesem Grund ist die fraktale Bildcodierung entstanden, die versucht, die Selbstähnlichkeit von fraktalen Bildern zu ihrer Codierung auszunutzen.

Diese neue Strategie wird manchmal mit bekannten Methoden der Bildkompression kombiniert. Pentland und Horowitz erläutern in [27] eine Methode, bei der ein Bild einer Wavelettransformation unterzogen wird und die drei Frequenzkanäle (horizontal, vertikal und nach beiden Richtungen) getrennt einer Vektorquantisierung unterworfen werden. Dabei wird Selbstähnlichkeit

der Frequenzbilder unterschiedlicher Auflösung ausgenutzt, um die Codierung der Vektorquantisierung zu optimieren. Für diese Methode wird eine Kompressionsrate von 13.3:1 bei einem Signal-/Rauschverhältnis von 30 dB als typisch angegeben.

Jacquin [14] stellt eine Methode vor, bei der die Ähnlichkeit von verschiedenen Blöcken eines Bildes ausgenutzt wird. Das Originalbild wird einmal in Quadrate, sogenannte domain cells, und einmal in kleinere Quadrate, die range cells, unterteilt. Nun wird für jede range cell eine geeignete Funktion gesucht, die eine domain cell auf diese range cell abbildet. Dabei soll die range cell möglichst gut approximiert werden. Es werden Operationen wie Kontrastskalierung, Invertierung, aber auch Drehung sowie Spiegelung um die mittleren Achsen auf die domain cells angewendet. Um für die range cells schneller geeignete domain cells zu finden, werden domain cells und range cells in drei Kategorien eingeteilt:

1. shade blocks, das sind Blöcke ohne Kontrast oder Textur
2. midrange blocks, das sind Blöcke mit feiner Textur
3. edge blocks, das sind Blöcke mit scharfen Kontrasten

Die shade blocks werden aus der Menge der domain cells entfernt, weil einfarbige Blöcke über Grauwertoperationen aus jedem beliebigen Block erzeugt werden können. Bei range cells der Klassen midrange oder edge blocks werden jeweils nur domain cells aus derselben Kategorie betrachtet. Wenn für jede range cell eine Transformation gefunden ist, wird die Vereinigung der Transformationen erzeugt. Der Fixpunkt der resultierenden Funktion approximiert das Ausgangsbild.

Jacquin gibt an, daß die Approximation bis auf Verlust feiner Textur und leichte Blockbildung sehr gut ausfällt. Die Codierung des Bildes „Lena“ ergibt eine Kompressionsrate von 0.68 bit / Pixel bei einem Signal-/Rauschverhältnis von 27.7 dB.

Ein vielversprechendes Mittel zur Ausnutzung der Selbstähnlichkeit fraktaler Bilder zu deren Codierung sind aufgrund ihrer Beschaffenheit die Iterierten Funktionensysteme (IFS). Eine anschauliche Erklärung dieses Typs von Fraktalen folgt im nächsten Abschnitt.

1.2 Eine Einführung in Iterierte Funktionensysteme

Hier wird eine kurze Erklärung von IFS gegeben, wie man sie in [2, 26] findet. Die Grundlagen von IFS werden z.B. in [13, 2, 3] ausführlich dargelegt.

Ein IFS ist eine Menge von Funktionen. Die Wirkungsweise dieser Funktionen läßt sich mit einem Gerät veranschaulichen, das wohl jedem vertraut ist: einer Kopiermaschine, die verkleinern kann. Jede Konfiguration einer solchen Maschine entspricht einer Funktion, die in einem IFS verwendet werden kann. Die einzige Bedingung dabei ist, daß die von der Kopiermaschine erzeugte Kopie eines Bildes, also der Output der Kopiermaschine, verkleinert ist.

Wird nun dieser Output als Input für die Maschine verwendet, der resultierende Output wieder als Input usw., zeigt sich folgendes: Die Kopien des Bildes werden kleiner und kleiner, bis

früher oder später (abhängig vom Verkleinerungsfaktor) nur noch ein einziger schwarzer Punkt zu sehen ist. Dieses Endresultat ergibt sich immer, egal welches Ausgangsbild ursprünglich als Input für die Kopiermaschine verwendet wurde.

Weit interessantere und vielfältigere Endresultate zeigen sich, wenn die Kopiermaschine zu einer „Multi Contraction Copy Machine“ (MCCM) [26] umgebaut wird:

Statt eines Linsensystems zur Verkleinerung werden nun mehrere unabhängige Linsensysteme verwendet, in Analogie zu IFS, die auch aus mehreren Funktionen bestehen. Sie alle verkleinern das Input–Bild und kopieren es auf das Output–Bild, und zwar abhängig von bestimmten Faktoren: Der erste Parameter der MCCM bestimmt die Anzahl der Linsensysteme, die verwendet werden sollen. Der zweite Parameter hält den Verkleinerungsfaktor jedes einzelnen Linsensystems fest, und der dritte beschreibt die Position des Ergebnisses von jedem einzelnen Linsensystem auf dem Output–Bild.

Betätigt man die MCCM einmal, wird das Input–Bild mit verschiedenen Verkleinerungsfaktoren kopiert und die Ergebnisse in einem bestimmten Muster auf dem Output–Bild angeordnet. Das Output–Bild der MCCM wird wieder als Input–Bild verwendet usw., wie im obigen Experiment für die Kopiermaschine. War bei dieser das Endresultat ein schwarzer Punkt, so ist bei der MCCM das Endresultat ein fraktales Bild, dessen Aussehen nur von den drei Parametern der MCCM abhängt.

Als Beispiel soll eine MCCM mit drei aktiven Linsensystemen genommen werden, die das Input–Bild jeweils um den Faktor $\frac{1}{2}$ verkleinern und die Ergebnisse in Form eines gleichseitigen Dreiecks anordnen. Abb. 1.2 zeigt die Anwendung der MCCM auf verschiedene Ausgangsbilder. Man sieht, daß das Endresultat immer das gleiche sein wird, unabhängig vom Ausgangsbild.

Das Ausgangsbild in der letzten Zeile zeigt spezielles Verhalten; es ändert sich nicht, egal wie oft die MCCM darauf angewendet wird. Diese Eigenschaft besitzt genau ein Bild, nämlich das Endresultat, das von allen Ausgangsbildern durch Iteration erreicht wird. Das Endresultat der MCCM von Abb. 1.2 ist das sogenannte Sierpinski – Dreieck [2].

Nachdem die Funktionsweise einer MCCM beschrieben wurde, kann die Analogie zu IFS gezogen werden. Die IFS, die in dieser Arbeit behandelt werden, bestehen aus endlich vielen kontraktiven affinen Transformationen. Man kann sich die Effekte dieser Transformationen auf geometrischer Ebene vorstellen: Sie beschreiben Rotation, Skalierung, Spiegelung und daraus zusammengesetzte Operationen. Jede Konfiguration der letzten beiden Parameter einer MCCM entspricht einer solchen Transformation[1].

Es werden alle Transformationen auf eine Ausgangsmenge angewendet; die einzelnen Ergebnisse werden vereinigt, wie bei der MCCM die verschiedenen Linsensysteme zusammen ein Output–Bild produzieren. Auf dieses Ergebnis werden wiederum alle Transformationen angewendet usw.

Analog zur MCCM produziert auch ein IFS unabhängig vom Ausgangsbild ein Endresultat, den sogenannten Attraktor. Dieser Attraktor ist eine Menge, die von dem IFS generiert wird.

[1] Umgekehrt gilt dies jedoch nicht, da die Skalierung, die eine affine Transformationen durchführt, nicht in allen Richtungen gleichförmig sein muß.

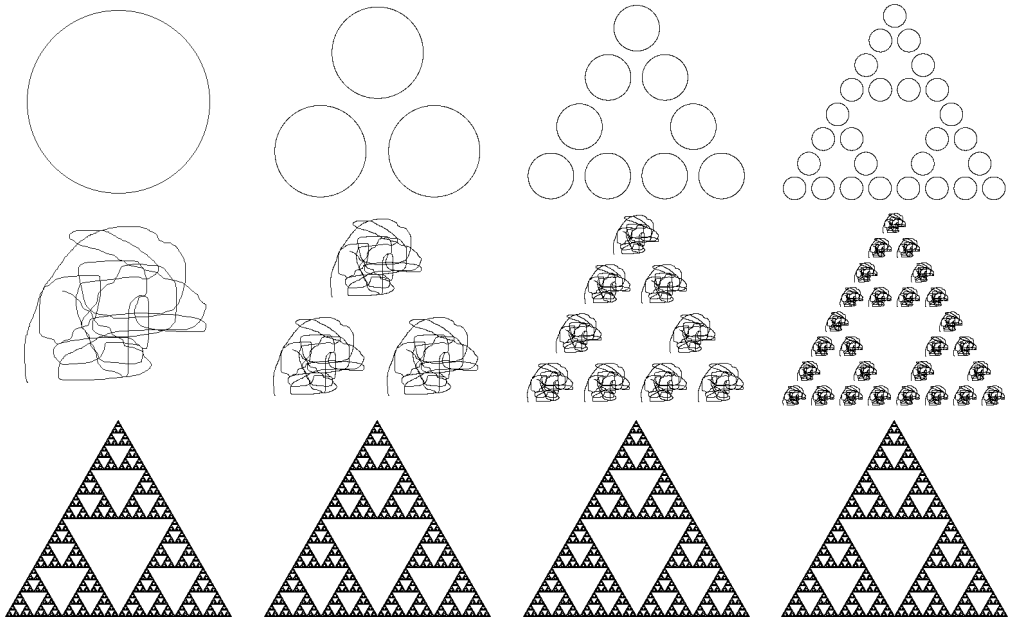


Abbildung 1.2: Die ersten drei Kopien verschiedener Ausgangsbilder einer MCCM

Nimmt man den Attraktor als Ausgangsmenge, ändert er sich nicht, egal wie oft die Transformationen des IFS darauf angewendet werden.

Jede Transformation eines IFS läßt sich mit sechs reellen Zahlen beschreiben. Das erklärt den geringen Speicherbedarf von Bildern, die durch ein IFS codiert sind.

1.3 Das inverse Problem

Bisher war von IFS die Rede, die abhängig von den im Abschnitt 1.2 erwähnten Parametern fraktale Bilder erzeugen. In der Praxis liegt der Fall meist anders: Es soll ein bereits vorhandenes Bild codiert werden. Dazu muß ausgehend vom gegebenen Bild ein IFS gefunden werden, dessen Attraktor das Bild möglichst gut approximiert. Diese Aufgabe ist bekannt als das inverse Problem.

Die Schwierigkeit dabei ist, daß ausgehend von den Parametern eines IFS keine Vorhersagen über die Form des zugehörigen Attraktors gemacht werden können. Es ist auch nicht offensichtlich, wie die Parameter eines IFS beschaffen sind, wenn nur der Attraktor dieses IFS gegeben ist.

Das inverse Problem ist nicht leicht zu lösen; es ist bis jetzt keine automatische vollständige Lösung publiziert. Es gibt jedoch Ansätze von einigen Autoren, auf die näher eingegangen werden soll.

Barnsley gibt an, mit IFS tatsächlich Kompressionsraten von 10000:1 erreichen zu können [5]. Er hat jedoch bis jetzt nur diese Resultate und nicht den Lösungsweg veröffentlicht.

Barnsley verwendet wie einige andere Autoren das Collage–Theorem. Es besagt in Worten ausgedrückt folgendes: Wird eine geeignete Überdeckung (Collage) eines gegebenen Bildes mit affinen Bildern [24, 7] seiner selbst gefunden, dann kann das Bild durch ein IFS approximiert werden. Abb. 1.3 zeigt ein Beispiel am Bild eines Blattes.

Es wird nun eine mathematische Beschreibung des Collage–Theorems gegeben. Die Sätze und Definitionen, auf denen sie aufbaut, sind im 2. Kapitel dieser Arbeit beschrieben.

Sei (X, d) ein vollständiger metrischer Raum mit Distanzfunktion d. Sei $L \in \mathcal{H}(X)$ eine beliebige Menge und $\epsilon > 0$ gegeben. Man wähle ein IFS mit Funktionen $\{w_1, \ldots, w_N\}$ und Kontraktionsfaktor $c = \max(c_i, i = 1, 2, \ldots, N)$, sodaß

$$d\left(L, \bigcup_{i=1}^{N} w_i(L)\right) \leq \epsilon. \tag{1.1}$$

Dann ist

$$d(L, \mathcal{A}) \leq \frac{\epsilon}{1-c} \tag{1.2}$$

wobei $\mathcal{A}$ der Attraktor des IFS ist.

ϵ beschreibt die Genauigkeit der Überdeckung. Bei komplexen Formen können beliebig viele Funktionen zur Collage verwendet werden, was ϵ entsprechend klein werden läßt. Verwendet man Funktionen mit kleinen Kontraktionsfaktoren c_i, bleibt der Kontraktionsfaktor c des IFS klein und damit die Approximation des Bildes gut.

Abbildung 1.3: Collage: Überdeckung eines Bildes mit affinen Bildern seiner selbst

Das Collage–Theorem wird auch von Horn verwendet, der in [12] ein interaktives System zur Ermittlung eines IFS für die Codierung eines gegebenen Bildes vorstellt. Der Benutzer erzeugt mit Parallelogrammen eine Überdeckung des gegebenen Bildes. Dabei wird ausschließlich der sogenannte meta–skew verwendet, mit dem sich sämtliche affinen Abbildungen für die Parallelogramme nachvollziehen lassen:

Wenn der Benutzer mit der Maus eine Kante eines Parallelogramms bewegt, folgen drei Kanten der Bewegung, während sich die vierte Kante so bewegt, daß die Form eines Parallelogramms erhalten bleibt.

Jedes Parallelogramm steht für eine Transformation. Die Parameter einer Transformation werden automatisch aus den Koordinaten des zugehörigen Parallelogramms ermittelt.

Sobald die Parallelogramme verändert wurden, stellt das System den Attraktor des resultierenden IFS dar. So kann der Benutzer sukzessive ein dem Original immer ähnlicheres Bild erzeugen.

Ein ebenfalls halbautomatischer Ansatz zur Lösung des inversen Problems stammt von den Autoren Libeskind–Hadas und Maragos in [19]. Sie benutzen die Operationen Erosion und Dilation aus dem Gebiet der mathematischen Morphologie [28]. Zunächst wird gezeigt, daß das Collage–Theorem auch für die Vereinigung von IFS mit einem fixen Teilbild Gültigkeit hat. Es sollen nun die selbstähnlichen Teilbilder des binären Ursprungsbildes ermittelt werden; der Rest des Bildes wird zum fixen Teilbild erklärt. Für jedes Teilbild sind die Parameter Translation, Kontraktionsfaktor und Drehwinkel gefragt. Zu deren Ermittlung wird das diskrete Skelett des Bildes ermittelt [22]. Die einzelnen Parameter werden folgendermaßen gefunden: Sei das Bild $A \in \mathbb{R}^2$, wobei $\mathbb{R}$ die Menge der reellen Zahlen ist.

- Translation
  Es gibt im Zentrum des Skeletts von A markante Kreuzungspunkte, die sich im Zentrum der Teilbilder wiederholen. Der Benutzer muß den Kreuzungspunkt von A sowie den Kreuzungspunkt des zu erzeugenden Teilbildes markieren. Die Translation ist dann gleich dem Vektor vom Kreuzungspunkt von A zum Kreuzungspunkt des Teilbildes.
- Kontraktionsfaktor
  - In x–und y–Richtung gleicher Kontraktionsfaktor
    Beim Prozeß der Skelettierung werden Scheiben ermittelt, die in dem Sinne maximal sind, daß sie vollständig in A enthalten sind und von keiner anderen Scheibe in A vollständig überdeckt werden. Der Kontraktionsfaktor ist dann das Verhältnis des Radius der größten Scheibe im Teilbild zum Radius der größten Scheibe in A.
  - In x–und y–Richtung verschiedener Kontraktionsfaktor
    Hier werden bestimmte Skalierungen in x– und y–Richtung zur Verfügung gestellt. Alle ihre Kombinationen werden auf A angewendet und die Kontraktionsfaktoren desjenigen Resultates gewählt, das dem Teilbild am nächsten kommt.
- Drehwinkel
  - In x–und y–Richtung gleicher Kontraktionsfaktor
    Es werden nur einige fixe Drehwinkel in Betracht gezogen. A wird mit jedem dieser Drehwinkel auf das Teilbild transformiert. Die Transformation, deren Ergebnis dem Teilbild am ähnlichsten ist, wird festgehalten.
  - In x–und y–Richtung verschiedener Kontraktionsfaktor
    Der Benutzer markiert neben den erwähnten Kreuzungspunkten weitere markante Punkte, aus denen der Drehwinkel errechnet wird.

Nachdem alle selbstähnlichen Teilbilder behandelt wurden, muß das fixe Teilbild, das A ergänzt, gespeichert werden. Dies geschieht durch den Benutzer, der geeignete maximale Scheiben selektiert, die das fixe Teilbild überdecken.

Libeskind–Hadas und Maragos geben als Anwendungsgebiete für ihren Algorithmus die Datenkompression und die Animation (Generierung von Bildsequenzen durch geringfügige Änderung der Parameter des IFS) an.

Ein automatischer Ansatz stammt von Mazel und Hayes, die in [23] diskrete, eindimensionale Signale codieren. Sie präsentieren zwei Varianten:

Beim ersten Algorithmus wird nur die strikte Selbstähnlichkeit berücksichtigt. Es werden Stützpunkte aus der Menge der Signaldaten gewählt. Dann wird der am weitesten links liegende Stützpunkt als Anfangsstützpunkt genommen und für alle Endstützpunkte (das sind alle Stützpunkte rechts vom Anfangsstützpunkt) je eine Funktion ermittelt, die das gesamte Signal auf den Bereich zwischen Anfangs– und Endstützpunkt abbildet. Diejenige der Funktionen, die dem ursprünglichen Signal am ähnlichsten ist, wird festgehalten. Der nächste Anfangsstützpunkt ist der Endstützpunkt der gespeicherten Funktion. Dieser Vorgang wird solange wiederholt, bis der am weitesten rechts liegende Stützpunkt erreicht ist.

Das zweite Verfahren berücksichtigt die partielle Selbstähnlichkeit der Signale. Hier wird das Signal zweimal in jeweils gleich große Intervalle geteilt, einmal in Zielintervalle und einmal in Adreßintervalle. Dabei ist die Länge eines Adreßintervalls größer als die eines Zielintervalls. Nun werden alle Adreßintervalle so auf ein Zielintervall abgebildet, daß sich ihre Anfangs– und Endpunkte decken. Diejenige Funktion wird gespeichert, deren Bild dem Zielintervall am ähnlichsten ist. Das wird für alle Zielintervalle durchgeführt. Durch die Wahl der Adreß– und Zielintervalle ist die Kompressionsrate bei diesem Verfahren vorherbestimmt.

Mazel und Hayes geben an, daß bei Quantisierung in der zweiten Methode relativ wenig störende Effekte auftreten. Diese Art der Codierung ist nur für in gewissem Grade selbstähnliche Signale sinnvoll. In [23] werden u.a. Beispiele mit seismischen Daten, Elektrokardiogramm–Daten und Signalen, die von einer menschlichen Stimme stammen, gebracht; dabei wurde die zweite Methode verwendet. Das letzte Beispiel der obigen Aufzählung ergibt mit 6.4:1 die beste Kompressionsrate bei einem analytischen Signal-/Rauschverhältnis von 16.9 dB.

Die Autoren Levy–Vehel und Gagalowicz stellen in [17] eine Optimierungsmethode vor, bei der die Distanz zwischen dem gegebenen Bild und dem Attraktor minimiert wird. Zunächst werden verschiedene Distanzfunktionen betrachtet und schließlich die Hausdorff–Metrik als geeignetste Funktion verwendet, was jedoch einen speziellen Optimierungsalgorithmus erfordert. Werden die Anfangswerte günstig gewählt, sind die Resultate dieser Methode gut; ist das nicht der Fall, besteht die Gefahr, daß der Algorithmus in einem lokalen Minimum endet.

Die oben beschriebene Optimierungsmethode wird von den Autoren in [18] für Grauwertbilder erweitert, d.h. es wird zusätzlich zu jeder Transformation eine Wahrscheinlichkeit ermittelt. Die Anwendung sowohl auf klassische (d.h. künstliche) Fraktale wie auf „natürliche Fraktale“ (z.B. das Bild eines Ahornblattes) ergibt zufriedenstellende Resultate; die Methode hat aber immer noch den Nachteil vieler lokaler Minima in der Distanzfunktion.

Robuster ist der zweite Algorithmus in [18], der auf der Methode des simulated annealing basiert. Hier wird die Gesamtenergie eines Systems langsam reduziert, während eine Minimumsuche in Form einer Wanderung durch definierte Systemzustände durchgeführt wird, bis die Systemenergie klein genug ist. Der Algorithmus ist zwar langsamer als die vorher beschriebene Optimierungsmethode, aber dafür unabhängig von den Startwerten.

Vrscay geht in [30] einen ähnlichen Weg, er verwendet zur Minimumsuche jedoch genetische Algorithmen.

1.4 Notation

Dies ist ein Überblick der in dieser Arbeit verwendeten Notationen.

Notation	*Erläuterung*
$\mathbb{N}$	Die Menge der natürlichen Zahlen (inklusive 0).
$\mathbb{R}$	Die Menge der reellen Zahlen.
$\mathbb{R}_0^+$	Die Menge der reellen Zahlen größer gleich 0: $\{x \in \mathbb{R} \mid x \geq 0\}$
$\mathbb{R}^2$	Die Ebene.
$\mathbb{R}^d$	Der d–dimensionale Raum.
$[a, b]$	Abgeschlossenes reelles Intervall: $\{x \in \mathbb{R} \mid a \leq x \leq b\}$
$[a, b)$	Rechts halboffenes reelles Intervall: $\{x \in \mathbb{R} \mid a \leq x < b\}$
$:=$	Definitionsgleichung, „wird definiert als“
$\lfloor x \rfloor$	floor von $x \in \mathbb{R}$: $\lfloor x \rfloor := \max\{n \in \mathbb{Z} \mid n \leq x\}$
$\lceil x \rceil$	ceiling von $x \in \mathbb{R}$: $\lceil x \rceil := \min\{n \in \mathbb{Z} \mid n \geq x\}$
$\emptyset$	Die leere Menge.
$f^{\circ n}(x)$	n-fache Anwendung der Funktion f auf x: $f^{\circ n}(x) := f(f^{\circ n-1}(x))$ für $n \in \mathbb{N}^+$, $f^{\circ 0}(x) := x$

Kapitel 2

Grundlagen Iterierter Funktionensysteme

Es wird eine kurze Einführung in die grundlegende Theorie Iterierter Funktionensysteme gegeben, die zum Verständnis der späteren Kapitel notwendig erscheint. Eine solche Einführung ist auch bei Hutchinson in [13] und in erweiterter Form bei Barnsley in [2, 3] zu finden. In [26] haben Peitgen et al. Iterierte Funktionensysteme sehr anschaulich und gut verständlich dargelegt.

Notation und Definitionen folgen teilweise [2, 7]. Wenn nicht anders erwähnt, sind Beweise für die Sätze in [13, 2, 7] zu finden.

Sei (X, d) ein vollständiger metrischer Raum mit Distanzfunktion d.

Definition 2.1 Eine Zahl $c \in \mathbb{R}_0^+$ heißt *Kontraktionsfaktor* einer Abbildung w auf X genau dann, wenn

$$d(w(x), w(y)) \leq cd(x, y) \qquad \forall\, x, y \in X \tag{2.1}$$

und es kein $k \in [0, c)$ mit

$$d(w(x), w(y)) \leq kd(x, y) \qquad \forall\, x, y \in X \tag{2.2}$$

gibt.

Definition 2.2 Eine Abbildung w auf X heißt *kontraktiv* genau dann, wenn ihr Kontraktionsfaktor $c < 1$ ist.

Definition 2.3 Ein $x \in X$ heißt *Fixpunkt* der Abbildung $w : X \to X$ genau dann, wenn es die *Fixpunktgleichung* $w(x) = x$ erfüllt. Die Fixpunkte von w sind also genau jene Elemente von X, die unter w auf sich selbst abgebildet werden.

Definition 2.4 Für eine kontraktive Abbildung $w : X \to X$ heißt die Folge $\{x_n\}$, $n = 0, 1, 2, \ldots$, mit $x_0 \in X$ und $x_{n+1} = w(x_n)$ für $n \geq 0$, der *Orbit* von x_0 unter w.

Bisher wurden Elemente von X und ihre Bilder unter kontraktiven Abbildungen betrachtet. Von größerem Interesse sind jedoch spezielle Teilmengen von X und die auf diese Teilmengen definierten Abbildungen.

Definition 2.5 Sei $\mathcal{H}(X)$ die Klasse aller nicht leeren, kompakten Untermengen von X.

Sei A_δ die *δ-Umgebung* von $A \in \mathcal{H}(X)$, $\delta \in \mathbb{R}_0^+$. Diese besteht aus der Menge der Punkte, die innerhalb einer Distanz δ von A liegen, d.h.

$$A_\delta := \{x \in X \mid d(x,a) \leq \delta \text{ für mindestens ein } a \in A\}. \tag{2.3}$$

Definition 2.6 Die Abbildung $d_h : \mathcal{H}(X) \times \mathcal{H}(X) \to \mathbb{R}_0^+$ mit

$$d_h(A,B) := \inf\{\delta \in \mathbb{R}_0^+ \mid A \subseteq B_\delta \text{ und } B \subseteq A_\delta\} \tag{2.4}$$

für $A, B \in \mathcal{H}(X)$ wird *Hausdorffmetrik* genannt.

Satz 2.7 *Ist (X,d) ein vollständiger metrischer Raum, so ist $(\mathcal{H}(X), d_h)$ ebenfalls ein vollständiger metrischer Raum.*

Ein w kann nun zu einer Abbildung auf $\mathcal{H}(X)$ erweitert werden, wobei die Kontraktivität erhalten bleibt. Statt einzelner Punkte werden Mengen transformiert.

Satz 2.8 *Ist $w : X \to X$ eine kontraktive Abbildung auf (X,d) mit Kontraktionsfaktor c, dann ist $w : \mathcal{H}(X) \to \mathcal{H}(X)$ mit*

$$w(A) := \{w(a) \mid a \in A\} \qquad A \in \mathcal{H}(X) \tag{2.5}$$

eine kontraktive Abbildung auf $(\mathcal{H}(X), d_h)$ mit Kontraktionsfaktor c.

2.1 Iterierte Funktionensysteme

Auf der Basis der obigen Sätze und Definitionen können nun Iterierte Funktionensysteme definiert und ihre Eigenschaften diskutiert werden.

Definition 2.9 Ein *Iteriertes Funktionensystem* (kurz IFS) ist eine endliche Menge von zwei oder mehr kontraktiven Abbildungen auf X.

Ein solches IFS wird auch deterministisches IFS genannt; im Gegensatz dazu steht das probabilistische IFS, dessen Abbildungen mit gewissen Wahrscheinlichkeiten versehen werden.

Die im folgenden definierte Abbildung charakterisiert den Effekt eines IFS auf ein Element aus $\mathcal{H}(X)$.

Definition 2.10 Der *Hutchinsonoperator* $\mathbf{w} : \mathcal{H}(X) \to \mathcal{H}(X)$ eines IFS $\{w_i \mid i = 1, \ldots, n\}$ ist die Vereinigung der Bilder unter den n Abbildungen des IFS:

$$\mathbf{w}(A) := \bigcup_{i=1}^{n} w_i(A) \qquad A \in \mathcal{H}(X) \tag{2.6}$$

Die folgenden beiden Sätze sind in der Theorie der IFS von großer Bedeutung.

Satz 2.11 *Der Hutchinsonoperator* $\mathbf{w} : \mathcal{H}(X) \to \mathcal{H}(X)$ *eines IFS ist eine kontraktive Abbildung auf dem vollständigen metrischen Raum* $(\mathcal{H}(X), d_h)$. *Der maximale Kontraktionsfaktor der Abbildungen des IFS ist der Kontraktionsfaktor von* $\mathbf{w}$ *und wird Kontraktionsfaktor des zugehörigen IFS genannt.*

Satz 2.12 *Der Hutchinsonoperator* $\mathbf{w}$ *hat einen eindeutigen Fixpunkt* $\mathcal{A} \in \mathcal{H}(X)$. $\mathcal{A}$ *ist die eindeutige Lösung von*

$$A = \mathbf{w}(A). \tag{2.7}$$

Der Orbit $\{A_0, \mathbf{w}(A_0), \mathbf{w}^{\circ 2}(A_0), \ldots\}$ *jedes Anfangselements* $A_0 \in \mathcal{H}(X)$ *unter* $\mathbf{w}$ *konvergiert zu* $\mathcal{A}$.

Definition 2.13 Der Fixpunkt $\mathcal{A}$ des Hutchinsonoperators wird *Attraktor* des zugehörigen IFS genannt.

Zur Illustration der obigen Sätze und Definitionen werden nun einige Beispiele gezeigt. Obwohl der Schwerpunkt dieser Arbeit auf IFS in $\mathbb{R}$ liegt, werden hier hauptsächlich IFS in $\mathbb{R}^2$ verwendet, weil diese anschaulicher sind. Dabei wird der metrische Raum $\mathbb{R}^2$ mit dem euklidischen Abstand verwendet, da dieser leicht darstellbar ist und dem „natürlichen“ Abstandsbegriff entspricht. Die verwendeten Abbildungen sind kontraktive affine Transformationen. Auf affine Transformationen in $\mathbb{R}$ wird später noch genauer eingegangen. Um affine Transformationen anschaulich beschreiben zu können, wird eine Notation verwendet, die in [25] genau definiert wird. $\mathfrak{S}(c_x, c_y)$ beschreibt eine Skalierung in x–Richtung mit dem Faktor c_x und in y–Richtung mit dem Faktor c_y. $\mathfrak{R}(\alpha)$ beschreibt eine Drehung um den Winkel α.

Der Effekt des Hutchinsonoperators des IFS $\{w_1, w_2, w_3\}$ auf $\mathbb{R}^2$ mit

$$\begin{aligned} w_1(\bar{x}) &= \mathfrak{S}(0.5, 0.5)\bar{x} \\ w_2(\bar{x}) &= \mathfrak{S}(0.5, 0.5)\bar{x} + \binom{0.25}{\sqrt{3/16}} \qquad \bar{x} \in \mathbb{R}^2 \\ w_3(\bar{x}) &= \mathfrak{S}(0.5, 0.5)\bar{x} + \binom{0.5}{0} \end{aligned} \tag{2.8}$$

ist in Abb. 2.1 illustriert. Die Ausgangsmenge ist das graue Rechteck links oben in der Abbildung. Die einmalige Anwendung des Hutchinsonoperators darauf ergibt die Menge rechts daneben. Hier wird die Kontraktivität der Transformationen deutlich: Die Ausgangsfigur ist unter den Transformationen in x– und y–Richtung kleiner geworden. Wird in der Anwendung des Hutchinsonoperators fortgefahren, ist das Ergebnis eine immer bessere Approximation des

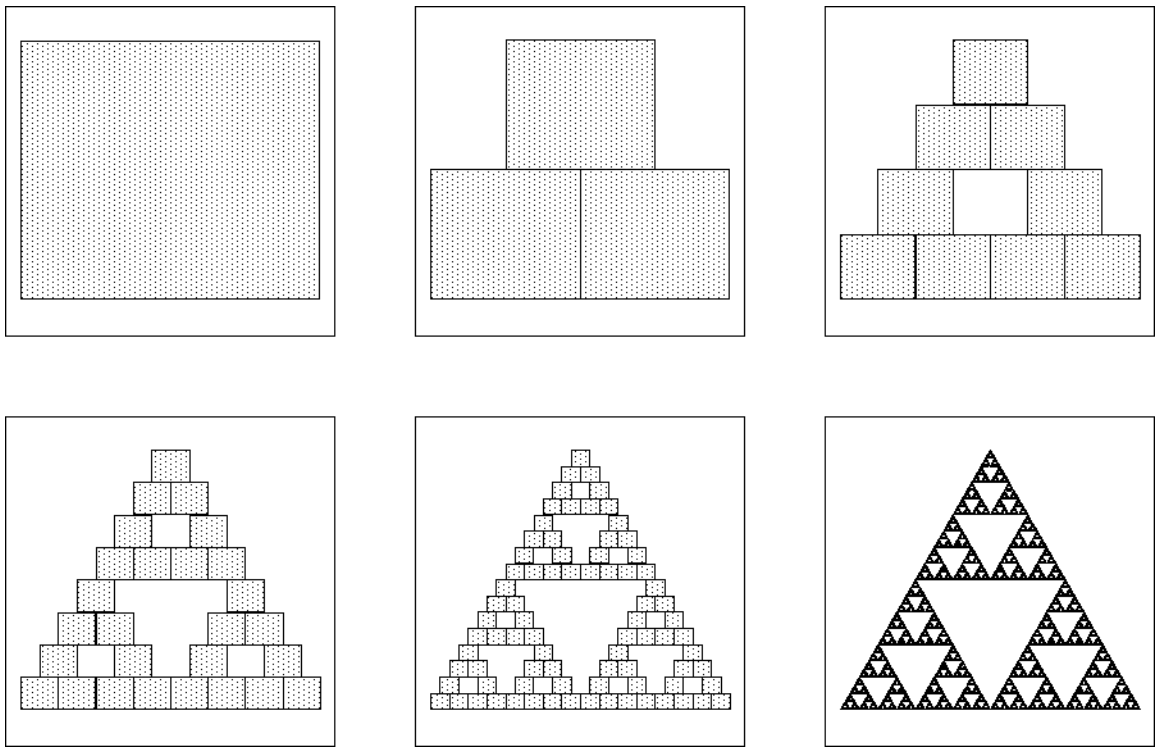
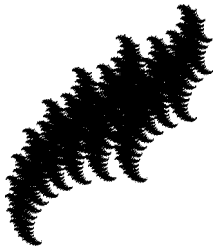


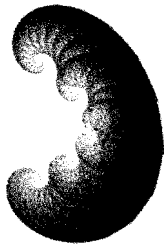
Abbildung 2.1: Wirkung des Hutchinsonoperators auf ein Rechteck

Attraktors, in diesem Fall des Sierpinski–Dreiecks. Das Sierpinski–Dreieck wird durch den Hutchinsonoperator dieses IFS immer angenähert, egal, welche Figur als Ausgangsbild genommen wird, und es ist der einzige Fixpunkt dieses Hutchinsonoperators.

Abb. 2.2 zeigt vier IFS in $\mathbb{R}^2$ und ihre Attraktoren. Das fünfte IFS liegt in $\mathbb{R}$; sein Attraktor wird als unterbrochene Linie dargestellt. Die Fixpunkte der Transformationen dieses IFS sind mit je einem Symbol $\star$ markiert.



$$w_1(\bar{x}) = \mathfrak{R}(10°)\mathfrak{S}(0.8, 0.8)\bar{x}$$
$$w_2(\bar{x}) = \mathfrak{R}(20°)\mathfrak{S}(0.64, 0.64)\bar{x} + \binom{5}{3}$$



$$w_1(\bar{x}) = \mathfrak{S}(0.9, 0.8)\mathfrak{R}(-20°)\bar{x}$$
$$w_2(\bar{x}) = \mathfrak{S}(0.8, 0.9)\mathfrak{R}(20°)\bar{x} + \binom{2}{1}$$

$$w_1(\bar{x}) = \mathfrak{S}(0.6, 0.6)\bar{x} + \binom{0.18}{0.36}$$
$$w_2(\bar{x}) = \mathfrak{S}(0.6, 0.6)\bar{x} + \binom{0.18}{0.12}$$
$$w_3(\bar{x}) = \mathfrak{S}(0.5, 0.5)\mathfrak{R}(-53°)\bar{x} + \binom{0.3}{0.45}$$
$$w_4(\bar{x}) = \mathfrak{S}(0.5, 0.5)\mathfrak{R}(53°)\bar{x} + \binom{0.33}{0.09}$$

$$w_1(\bar{x}) = \mathfrak{S}(0.5, 0.5)\bar{x}$$
$$w_2(\bar{x}) = \mathfrak{S}(0.5, 0.5)\bar{x} + \binom{0}{0.5}$$
$$w_3(\bar{x}) = \mathfrak{S}(0.25, 0.8)\mathfrak{R}(45°)\bar{x} + \binom{-0.7}{0}$$
$$w_4(\bar{x}) = \mathfrak{S}(0.25, 0.8)\mathfrak{R}(-45°)\bar{x} + \binom{0.7}{0}$$

$$w_1(\bar{x}) = \mathfrak{S}(0.5, 0.0)\bar{x}$$
$$w_2(\bar{x}) = \mathfrak{S}(0.35, 0.0)\bar{x} + \binom{0.65}{0.0}$$

Abbildung 2.2: Einige IFS mit zugehörigen Attraktoren

2.2 Affine Transformationen auf $\mathbb{R}$

Die im vorigen Abschnitt gezeigten IFS in $\mathbb{R}^2$ haben Attraktoren von erstaunlicher Komplexität. Die Attraktoren von IFS, die in $X = \mathbb{R}$ mit dem euklidischen Abstand und affinen Transformationen definiert sind, kurz IFS in $\mathbb{R}$, sind bei weitem einfacher. Auch die Vielfalt von affinen Transformationen in $\mathbb{R}$ ist stark begrenzt. Während zu affinen Transformationen in $\mathbb{R}^2$ Drehung, Schärung, Spiegelung, Skalierung und Translation gehören, kann eine affine Transformation in $\mathbb{R}$ höchstens aus Skalierung, Spiegelung und Translation bestehen.

Es werden nun die Eigenschaften von affinen Transformationen in $\mathbb{R}$ untersucht. Basis hiefür ist die Betrachtung von affinen Transformationen auf $\mathbb{R}^d$ in Kapitel 3 von [25]. Dort finden sich die nötigen Beweise bzw. Referenzen.

Definition 2.14 Eine *affine Transformation* $w : \mathbb{R} \to \mathbb{R}$ ist eine Abbildung, die jedem $x \in \mathbb{R}$

$$w(x) := kx + d \tag{2.9}$$

zuordnet, wobei k und $d \in \mathbb{R}$.

2.2.1 Kontraktionsfaktor einer affinen Transformation auf $\mathbb{R}$

Sei $w(x) = kx + d$ eine affine Transformation auf $\mathbb{R}$. $|k|$ ist der Kontraktionsfaktor von w.

w ist genau dann kontraktiv, wenn $k \in (-1, 1)$.

2.2.2 Fixpunkt einer kontraktiven affinen Transformation auf $\mathbb{R}$

Sei $w(x) = kx + d$ eine kontraktive affine Transformation auf $\mathbb{R}$. Nach Definition 2.3 ist $x_F \in \mathbb{R}$ genau dann Fixpunkt von w, wenn x_F die Fixpunktgleichung $w(x_F) = x_F$ erfüllt. Der Fixpunkt x_F von w kann demnach leicht errechnet werden:

$$x_F = \frac{d}{(1-k)} \tag{2.10}$$

x_F ist der einzige Fixpunkt von w.

2.2.3 Orbit einer kontraktiven affinen Transformation auf $\mathbb{R}$

Hier kann man generell die zwei Fälle einer affinen Transformation mit oder ohne Spiegelung unterscheiden. Sei $w(x) = kx + d$ eine kontraktive affine Transformation auf $\mathbb{R}$. Hat k positives Vorzeichen, dann ist w eine Transformation ohne Spiegelung. Abb. 2.3 zeigt ein Beispiel für den Orbit einer kontraktiven affinen Transformation w auf $\mathbb{R}$ ohne Spiegelung. w wird auf den Startpunkt in der Abbildung angewendet; der Startpunkt ist durch einen Bogen mit dem resultierenden Punkt verbunden. Auf diesen Punkt wird wiederum w angewendet etc. bis der Fixpunkt angenähert ist.

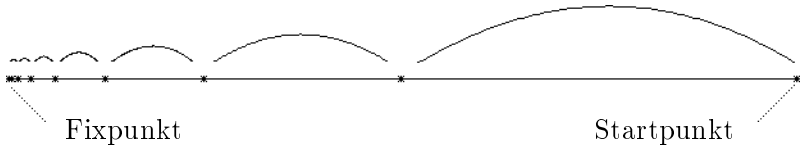


Abbildung 2.3: Orbit einer kontraktiven affinen Transformation auf $\mathbb{R}$ ohne Spiegelung

Hat k negatives Vorzeichen, dann ist w eine Transformation mit Spiegelung. Abb. 2.4 zeigt ein Beispiel für den Orbit einer kontraktiven affinen Transformation auf $\mathbb{R}$ mit Spiegelung.

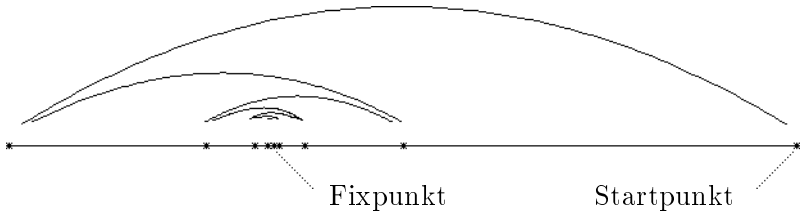


Abbildung 2.4: Orbit einer kontraktiven affinen Transformation auf $\mathbb{R}$ mit Spiegelung

2.2.4 Eine alternative Darstellung für kontraktive affine Transformationen auf $\mathbb{R}$

Sei $w(x) = kx+d$ eine kontraktive affine Transformation auf $\mathbb{R}$. Aus Abschnitt 2.2.2 ist bekannt, daß eine kontraktive affine Transformation $w(x) = kx+d$ einen eindeutigen Fixpunkt x_F besitzt. w kann nun als

$$w(x) = k(x - x_F) + x_F \tag{2.11}$$

dargestellt werden. Die Sätze 3.11 und 3.12 in [25] beweisen, daß man diese und die ursprüngliche Darstellung eindeutig ineinander überführen kann. Die Fixpunkt–Darstellung hat den Vorteil, daß der Fixpunkt von w sofort ersichtlich ist.

Kapitel 3

Diskrete Transformationen und diskreter Attraktor

Damit ein Attraktor auf dem Bildschirm eines Rechners dargestellt werden kann, muß er in ein digitales Bild umgewandelt werden. Dazu wird meist ein Algorithmus namens Chaos–Spiel [2] verwendet. Alternativ dazu wurden in [16] und ausführlich in [25] Algorithmen vorgestellt, die auf der Diskretisierung von Transformationen fußen und die Darstellung von Attraktoren effizienter gestalten. Da die Betrachtung des inversen Problems in den nachfolgenden Kapiteln auf den letztgenannten Ergebnissen beruht, werden diskrete Transformationen sowie diskrete Attraktoren hier kurz beschrieben.

3.1 Abtasten des Attraktors

Zunächst wird das Abtasten eines Attraktors in $\mathbb{R}^2$ beschrieben. Ein IFS in $\mathbb{R}^2$ definiert einen Attraktor, der eine kompakte Teilmenge von $\mathbb{R}^2$ ist. Diese Teilmenge kann nun wie folgt als Binärbild auf den Bildschirm gebracht werden: Der Bildschirm wird mit seiner Indexmenge identifiziert, die eine endliche Teilmenge von $\mathbb{N}^2$ ist. Der Teilbereich von $\mathbb{R}^2$, der den Attraktor enthält, wird in kleine Quadrate unterteilt, die mit den Bildschirmpixeln korrespondieren. Ein Pixel wird dann und nur dann auf 1 gesetzt, wenn das korrespondierende Quadrat mindestens einen Punkt des Attraktors enthält. Das resultierende Binärbild wird der *abgetastete Attraktor* genannt.

Attraktoren in $\mathbb{R}$ werden entsprechend abgetastet; der Teilbereich von $\mathbb{R}$, der den Attraktor enthält, wird dementsprechend in Intervalle geteilt.

Es folgt eine kurze Beschreibung des oben erwähnten Chaos–Spiels.

Chaosspiel(IFS $\{w_i \mid i = 1, \ldots, n\}$, $\bar{x}_0$):

- Der Anfangspunkt $\bar{x}_0$ muß in $\mathcal{A}$ liegen; dies ist z.B. für die Fixpunkte der Transformationen des IFS der Fall. j wird auf 0 gesetzt.
- **repeat**
  Das mit $\bar{x}_j$ korrespondierende Pixel wird gesetzt. Nun wird $\bar{x}_{j+1} = w_{i_j}(\bar{x}_j)$ berechnet, wobei jedes $i_j \in \{1, \ldots, n\}$ mit gleicher Wahrscheinlichkeit gewählt wird.
  until Qualität des Attraktorbildes ist gut genug.

Der Algorithmus terminiert nicht eindeutig; mit fortschreitender Zeit werden immer mehr Pixel des Attraktors wiederholt getroffen. Deshalb muß man ein Kriterium für die Qualität des resultierenden Bildes festlegen. Hier werden in der Praxis der Benutzer bzw. Erfahrungswerte entscheiden. Man bricht z.B. ab, wenn eine bestimmte Zeitlang kein Pixel mehr schwarz gesetzt wurde.

Da die ermittelte Punktfolge zufällig über dem Attraktor verteilt ist, wird der Attraktor durch diese Punktfolge regelmäßig überdeckt. Nach einer Anzahl von Iterationen, die deutlich größer als die Anzahl der Pixel des Attraktors ist, wird der Attraktor dicht genug überdeckt sein (Abschnitt 2.4 in [4]).

Attraktoren mit dem Chaos–Spiel abzutasten ist aufwendig, denn der Attraktor wird weit genauer berechnet als zur Darstellung notwendig. Dies ist deshalb der Fall, weil durch das Chaos–Spiel der Attraktor angenähert wird, der ja eine Teilmenge von $\mathbb{R}$ bzw. $\mathbb{R}^2$ ist. Da nicht offensichtlich ist, wie viele Iterationen für eine bestimmte Bildschirmauflösung genügen, muß die Zahl der Iterationen entsprechend hoch gewählt werden, sodaß viele Pixel öfter als einmal gesetzt werden.

3.2 Diskrete Transformationen

Der Attraktor eines IFS ist eine Teilmenge von $\mathbb{R}^2$. Um einen Attraktor für eine gegebene Bildschirmauflösung berechnen zu können, muß er als endliche Menge dargestellt werden. Es wird der diskrete Raum $\mathbb{P}^2 \subset \mathbb{N}^2$ der Pixel des Bildschirms verwendet. Der Einfachheit halber wird ein quadratischer Schirm angenommen. Für einen Bildschirm mit der Größe $R \in \mathbb{N}$ wird definiert:

$$\mathbb{P}_R := \{0, 1, \ldots, R-1\} \tag{3.1}$$

und

$$\mathbb{P}_R^2 := \mathbb{P}_R \times \mathbb{P}_R. \tag{3.2}$$

Man identifiziert den Bildschirm mit $[0,R)\times[0,R)\subset\mathbb{R}^2$; jedes Pixel $p=(p_x,p_y)\in\mathbb{P}_R^2$ repräsentiert ein Einheitsquadrat $[p_x,p_x+1)\times[p_y,p_y+1)\subset\mathbb{R}^2$. Von nun an wird $\mathbb{P}$ anstelle von $\mathbb{P}_R$ verwendet. Der Einfachheit halber wird angenommen, daß der Attraktor vollständig in $[0,R)\times[0,R)$ liegt, was aber keine Einschränkung darstellt, da jedes IFS in ein IFS transformiert werden kann, dessen Attraktor diese Bedingung erfüllt und der seine Form bis auf Verkleinerung und Verschiebung behält [2].

Da der Attraktor eines IFS eine Menge ist, muß eine Beziehung zwischen Teilmengen von $\mathbb{P}^2$ und Binärbildern hergestellt werden. Pixelmengen $I\in\mathcal{H}(\mathbb{P}^2)$ und Binärbilder sind isomorph zueinander, wenn die Pixel $\in I$ genau die gesetzten Pixel des Binärbildes sind. Deshalb ist es zulässig, Pixelmengen $I\in\mathcal{H}(\mathbb{P}^2)$ der Einfachheit halber als Bilder zu bezeichnen. Es muß auch die Bedeutung von $\mathbf{w}$ auf $\mathbb{P}^2$ definiert werden; dazu wird eine Verbindung zwischen $\mathbb{P}^2$ und $\mathbb{R}^2$ hergestellt: Man führt eine Abbildung $\varrho:\mathbb{P}^2\to\mathbb{R}^2$ ein, die für jedes Pixel diejenige Teilmenge von $\mathbb{R}^2$ angibt, durch die es repräsentiert wird. Jedes Pixel wird durch seinen Mittelpunkt repräsentiert.

$$\varrho(p):=\begin{pmatrix}p_x+0.5\\ p_y+0.5\end{pmatrix}\in\mathbb{R}^2 \qquad p\in\mathbb{P}^2 \tag{3.3}$$

Nach Anwendung einer affinen Transformation auf $\varrho(p)$ für ein Pixel p muß das Resultat wiederum nach $\mathbb{P}^2$ abgebildet werden. Dazu dient die Funktion $\Delta:\mathbb{R}^2\to\mathbb{P}^2$, die auf Punkten in $\mathbb{R}^2$ operiert. Δ liefert das Pixel, das einen gegebenen Punkt enthält.

$$\Delta(\bar{x}):=(\lfloor\bar{x}_1\rfloor,\lfloor\bar{x}_2\rfloor) \qquad \bar{x}=\tbinom{\bar{x}_1}{\bar{x}_2}\in\mathbb{R}^2 \tag{3.4}$$

Es ist sicher, daß das Resultat von Δ innerhalb von $\mathbb{P}^2$ liegt; alles außerhalb des Bildschirms wird verworfen.

Für eine affine Transformation $w:\mathbb{R}^2\to\mathbb{R}^2$ ist ihre zugehörige *diskrete Transformation* $w':\mathbb{P}^2\to\mathbb{P}^2$ wie folgt definiert:

$$w'(p):=(\Delta\circ w\circ\varrho)(p) \qquad p\in\mathbb{P}^2 \tag{3.5}$$

Das Pixel p wird zuerst mittels ϱ nach $\mathbb{R}^2$ abgebildet, dann wird die Transformation w angewendet, und die resultierende Teilmenge von $\mathbb{R}^2$ wird schließlich durch Δ in eine Menge von Pixeln transformiert. In Abb. 3.1 wird gezeigt, wie ein Pixel auf ein anderes abgebildet wird. Das strichlierte Parallelogramm zeigt, wie das gesamte Pixel nach der Transformation durch w aussehen würde.

3.3 Diskreter Attraktor

Der diskrete Hutchinsonoperator $\mathbf{w}':\mathcal{H}(\mathbb{P}^2)\to\mathcal{H}(\mathbb{P}^2)$ eines IFS $\{w_i\mid i=1,\ldots,n\}$ bildet eine Pixelmenge auf eine Pixelmenge ab und ist folgendermaßen definiert:

$$\mathbf{w}'(I):=\bigcup_{i=1}^{n}\bigcup_{p\in I}\{w_i'(p)\} \qquad I\in\mathcal{H}(\mathbb{P}^2) \tag{3.6}$$

Jedes Pixel des Bildes I wird durch jede Transformation des IFS diskret transformiert.

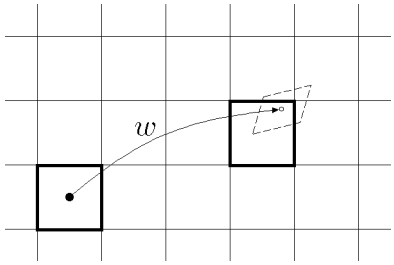


Abbildung 3.1: Eine diskrete Transformation

Definition 3.1 Hat man ein IFS mit diskretem Hutchinsonoperator $\mathbf{w}'$, so wird jede Menge $\mathcal{A}_d \in \mathcal{H}(\mathbb{P}^2)$ mit

$$\mathcal{A}_d = \mathbf{w}'(\mathcal{A}_d) \tag{3.7}$$

ein *diskreter Attraktor* genannt.

Der diskrete Attraktor ist nicht eindeutig, wie es beim tatsächlichen Attraktor der Fall ist (Satz 2.12), da durch die Diskretisierung $\mathbf{w}'$ im allgemeinen keine kontraktive Transformation mehr ist.

Es gibt jedoch nur eine endliche Anzahl von diskreten Attraktoren für ein IFS. Außerdem existiert für jedes IFS genau eine maximale Lösung $\mathcal{A}_d^\bullet$, die alle diskreten Attraktoren beeinhaltet. In [25] finden sich zwei Algorithmen, die einen diskreten bzw. den maximalen diskreten Attraktor eines IFS berechnen.

Unterschied zwischen abgetastetem und diskretem Attraktor

Es stellt sich die Frage, inwieweit ein diskreter Attraktor vom abgetasteten Attraktor desselben IFS differiert. Des weiteren ist interessant, ob man diese Differenz beschränken kann.

Abb. 3.2 zeigt Abbildungen des Farn–Attraktors aus [2] mit einer Auflösung von 256×256 Pixeln. Das linke Bild wurde mit dem in Abschnitt 3.1 beschriebenen Chaos–Spiel erzeugt. Das rechte Bild stellt einen diskreten Attraktor dar. Es sind geringfügige Unterschiede zwischen den beiden Bildern zu sehen.

Basierend auf [29] ist in [25] bewiesen, daß eine Beschränkung für die Differenz zwischen jedem diskreten Attraktor und dem abgetasteten Attraktor desselben IFS existiert.

Als Maß für die Differenz wird die Haussdorff–Metrik (Def. 2.6) verwendet. Dazu wird der diskrete Attraktor eindeutig mit einem Element aus $\mathcal{H}(\mathbb{R}^2)$ identifiziert.

Sei $\mathbf{w}$ der Hutchinsonoperator eines IFS mit Kontraktionsfaktor c und Attraktor $\mathcal{A}$, dann gilt

$$d_h(\mathcal{A}, \mathcal{A}_d) \leq \frac{1/\sqrt{2}}{1-c} \tag{3.8}$$

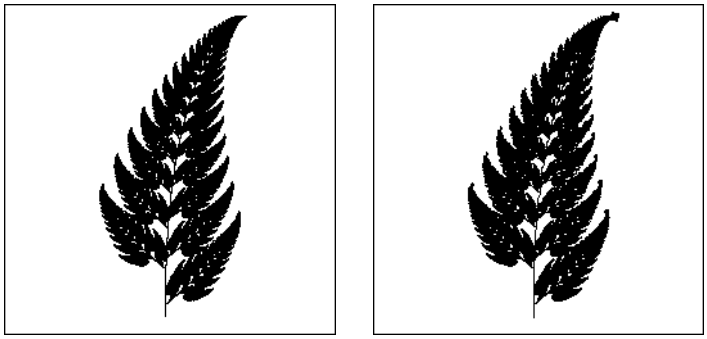


Abbildung 3.2: Ein abgetasteter und ein diskreter Farn–Attraktor

für jeden diskreten Attraktor $\mathcal{A}_d \in \mathcal{H}(\mathbb{P}^2)$. Daraus ist ersichtlich, daß die Differenz zwischen jedem $\mathcal{A}$ und $\mathcal{A}_d$ nur vom Kontraktionsfaktor c des betreffenden IFS abhängt.

Für ein IFS auf $\mathbb{R}$, gilt:

$$d_h(\mathcal{A}, \mathcal{A}_d) \leq \frac{1/2}{1-c}. \tag{3.9}$$

Kapitel 4

Der Algorithmus MatchRatio

Im Kapitel 3 wurde kurz der Weg vom IFS zum diskreten Attraktor beschrieben. Nun soll der umgekehrte Vorgang im eindimensionalen Fall betrachtet werden, also das eindimensionale inverse Problem.

Im folgenden wird der Algorithmus MatchRatio vorgestellt, der für eine beliebige binäre Pixelzeile ein IFS findet, für das die Pixelzeile P ein diskreter Attraktor ist. Es wird die in Kapitel 3 erklärte Mittelpunktrepräsentation verwendet.

MatchRatio besteht aus zwei Teilen, die logisch getrennt sind, jedoch verschachtelt ausgeführt werden. Der eigentliche Algorithmus wird in diesem Kapitel beschrieben: Die Verwendung einer invarianten Eigenschaft der Pixelzeile wird gezeigt; es folgt die Beschreibung des Matching, eines wichtigen Bestandteils von MatchRatio. Der nächste Teil des Kapitels ist der globalen Strategie von MatchRatio gewidmet. Schließlich werden einige Eigenschaften von MatchRatio betrachtet und Experimente beschrieben.

Der zweite Teil von MatchRatio, die Auffindung von Parametern für die möglichen Transformationen, falls sie existieren, wird im Kapitel 5 vorgestellt.

4.1 Die Ratios – ein invariantes Merkmal

Zunächst werden von MatchRatio Kandidaten für Transformationen des gesuchten IFS ermittelt. Dazu wird ein Merkmal der Pixelzeile P verwendet, und zwar die Invarianz des Verhältnisses der Längen von schwarzen und weißen Zusammenhangskomponenten von P. (Eine Zusammenhangskomponente von P ist eine Folge von gleichfarbigen Pixeln in P, die links und rechts von andersfarbigen Pixeln bzw. von Anfang oder Ende von P begrenzt ist.)

Definition 4.1 Ein *Dash* von P ist eine schwarze Zusammenhangskomponente von P. Ein *Gap* von P ist eine weiße Zusammenhangskomponente von P.

Satz 4.2 *Das Verhältnis der Längen von Dashes und Gaps ist in* $\mathbb{R}$ *unter affinen Transformationen invariant.*

Beweis: Sei $w = cx + d$ eine affine Transformation in $\mathbb{R}$. Gegeben ist ein Dash der Länge d und ein Gap der Länge g. Das Verhältnis r_1 der Längen von Dash und Gap ist

$$r_1 = \frac{d}{g}.$$

Transformiert man Dash und Gap mit w, dann ist die Länge des Dashes cd und die Länge des Gaps cg. Das Verhältnis r_2 der Längen von Dash und Gap nach der Transformation ist

$$r_2 = \frac{cd}{cg}.$$

Durch Kürzen sieht man, daß

$$\frac{d}{g} = \frac{cd}{cg}$$

Es gilt also $r_1 = r_2$.

□

Die Invarianz wird jedoch gestört, da wir uns im diskreten Raum $\mathbb{P}$ der Pixel befinden. Denn eine Zusammenhangskomponente wird in $\mathbb{P}$ durch ein diskretes w' transformiert, wie das im Abschnitt 3.2 beschrieben wurde. Das heißt, daß die Zusammenhangskomponente erst durch die Funktion ϱ von $\mathbb{P}$ auf $\mathbb{R}$ abgebildet wird, dann mit einem w transformiert und zuletzt durch die Funktion Δ wieder auf $\mathbb{P}$ abgebildet wird. Die Länge vom Ergebnis eines w in $\mathbb{R}$ wird sich durch die Abbildung Δ auf $\mathbb{P}$ meist ändern. Überdies kann sie abhängig von ihrer Position in $\mathbb{P}$ durchaus verschiedene Längen erhalten. Durch diese Änderung der Längen von Zusammenhangskomponenten verändern sich natürlich auch die zugehörigen Ratios.

Es kann jedoch ein Intervall für die möglichen Längen einer Zusammenhangskomponente unter einem w' mit Kontraktionsfaktor c angegeben werden. Das Intervall $\mathcal{D}(c, d)$ für die möglichen Längen eines Dashes d unter w' ist z.B.

$$\mathcal{D}(c, d) = [\lfloor c(d-1) \rfloor + 1, \lceil c(d-1) \rceil + 1].$$

In den Abschnitten 4.2.1 und 4.2.2 werden die Intervalle für Dashes und Gaps genauer beschrieben. Aus diesen wird dann ein Intervall für die möglichen Ratios dieser Längen, das sogenannte *Ratio–Intervall*, ermittelt. Ein grundlegendes Prinzip von MatchRatio ist es, die Ratio–Intervalle mit den Ratios der ursprünglichen Pixelzeile P zu vergleichen und so zu bestimmen, ob w' mit solchen Kontraktionsfaktoren Teil des gesuchten IFS sein können. Dabei muß berücksichtigt werden, daß für Kontraktionsfaktoren, die klein genug sind, weiße Zusammenhangskomponenten „verschwinden“ können. Das passiert dann, wenn eine weiße Zusammenhangskomponente unter einem w in $\mathbb{R}$ so klein wird, daß für diese unter Δ kein weißes Pixel mehr entsteht.

Der genaue Vorgang des Vergleichens der Ratios von P und der Ratio–Intervalle von P unter gewissen Kontraktionsfaktoren c, das sogenannte *Matching*, wird später dargestellt.

Hier genügt folgender Schluß: Sind die Ratios von P in den Ratio–Intervallen von P unter bestimmten Kontraktionsfaktoren enthalten, dann *können* Transformationen mit solchen Kontraktionsfaktoren Teil des gesuchten IFS sein. Diese Transformationen werden mögliche Transformationen genannt. Wegen der Beteiligung von Intervallen an dem Auswahlprozeß ist es nicht möglich, hier sichere Aussagen zu machen.

Ein w für jede dieser möglichen Transformationen wird, sofern es existiert, im zweiten Teil von MatchRatio ermittelt.

4.2 Die Ratio–Intervalle

Zur Ermittlung der Ratio–Intervalle müssen erst die Intervalle der möglichen Längen von Dashes und Gaps von P unter diskreten w' festgehalten werden.

Es wird wiederum die Mittelpunktrepräsentation verwendet und zur Ermittlung der Intervalle der möglichen Längen von Dashes und Gaps eine Transformation $w = cx + d$ ohne Spiegelung mit fixem, aber beliebigem c und variablem d herangezogen. Im Fall einer Transformation mit Spiegelung ist statt c der Betrag $|c|$ von c zu verwenden. Die Pixelbreite in $\mathbb{P}$ sei 1.

4.2.1 Das Intervall für die Dashes

Definition 4.3 Das Intervall $\mathcal{D}(c, d)$ beschreibt die mögliche Länge eines Dashes d unter einer diskreten Transformation w mit Kontraktionsfaktor c.

Ein Dash habe die Länge d, was auch der Anzahl seiner Pixel entspricht. Wegen der Mittelpunktrepräsentation ist der Abstand $d - 1$ vom ersten bis zum letzten Pixelmittelpunkt des Dashs relevant. Nach der Transformation mit w in $\mathbb{R}$ beträgt dieser Abstand $c(d-1)$. Δ liefert nun zu jedem transformierten Pixelmittelpunkt das Pixel, in dem dieser enthalten ist. Es wird schwarz gefärbt. Zur Erinnerung sei erwähnt, daß ein Pixel i einen Punkt dann enthält, wenn dieser im Intervall $[i, i+1)$ enthalten ist. Es stellt sich also die Frage, wieviele Pixel man maximal bzw. minimal mit dem Abstand $u := c(d-1)$ vom ersten bis zum letzten Pixelmittelpunkt des Dashs unter w überdecken kann.

Die untere Grenze des Intervalls

Die Anzahl der Pixel, die u überdeckt, ist dann minimal, wenn u genau an der linken Grenze eines Pixels beginnt. Das heißt, daß durch den ersten Pixelmittelpunkt des transformierten Dashs gerade dieses Pixel schwarz gefärbt wird, aber nicht das links davon, obwohl der Dash in das letztere hineinragt.

Sei $r := u - \lfloor u \rfloor$. Wir unterscheiden zwei Fälle:

1. $r = 0$
   u ist ganzzahlig. Weil die Pixelbreite 1 beträgt, befindet sich der letzte Pixelmittelpunkt ebenfalls genau an der linken Grenze eines Pixels. Dieses ist das am weitesten rechts gelegene, das schwarz gefärbt wird. Damit werden wegen $u = \lfloor u \rfloor$ insgesamt $\lfloor u \rfloor + 1$ Pixel gefärbt. Dieser Fall ist links in der Abb. 4.1 illustriert. Oben ist der Abstand u vom ersten bis zum letzten Pixelmittelpunkt zu sehen, wobei diese Pixelmittelpunkte hervorgehoben sind; darunter ist das Ergebnis von Δ für diesen Dash zu sehen.

2. $r > 0$
   Der Abstand u wird gegenüber dem obigen Fall um r verlängert. Wegen $r < 1$ und Pixelbreite $= 1$ kann die nächste Pixelgrenze nie erreicht werden, es werden also $\lfloor u \rfloor + 1$ Pixel schwarz gefärbt. Die Abb. 4.1 rechts zeigt den Fall mit einem großen r.

Es werden also in allen Fällen minimal $\lfloor u \rfloor + 1$ Pixel schwarz gefärbt.

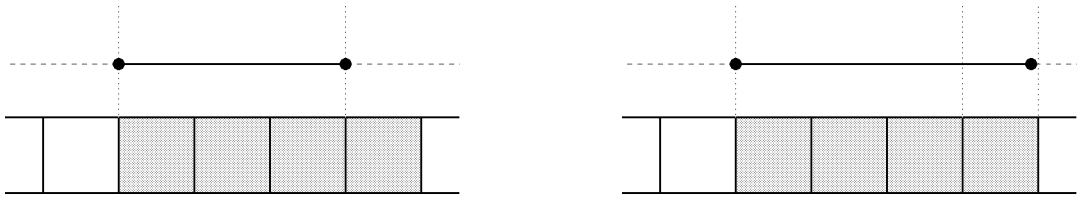


Abbildung 4.1: Es werden minimal $\lfloor u \rfloor + 1$ Pixel gefärbt

Die obere Grenze des Intervalls

Die Anzahl der Pixel, die u überdeckt, ist dann maximal, wenn u möglichst weit links in einem Pixel, aber nicht auf der linken Grenze des nächsten Pixels beginnt. Das heißt, daß durch den ersten Pixelmittelpunkt des transformierten Dashs das erste Pixel schwarz gefärbt wird, obwohl es der Dash nicht völlig überdeckt.

Wir unterscheiden wiederum zwei Fälle:

1. $r = 0$
   u ist ganzzahlig. Weil die Pixelbreite 1 beträgt, befindet sich der letzte Pixelmittelpunkt an derselben Stelle in einem Pixel wie der erste. Dieses ist das am weitesten rechts gelegene, das schwarz gefärbt wird. Damit werden wegen $u = \lceil u \rceil$ insgesamt $\lceil u \rceil + 1$ Pixel gefärbt. Dieser Fall ist links in der Abb. 4.2 illustriert.

2. $r > 0$
   Der Abstand u wird gegenüber dem obigen Fall um r verlängert. Wegen $r > 0$ wird die nächste Pixelgrenze sicher überschritten;es wird also ein Pixel mehr als im ersten Fall schwarz gefärbt. Das ergibt $\lceil u \rceil + 1$ schwarze Pixel. Ein Beispiel ist rechts in der Abb. 4.2 zu sehen.

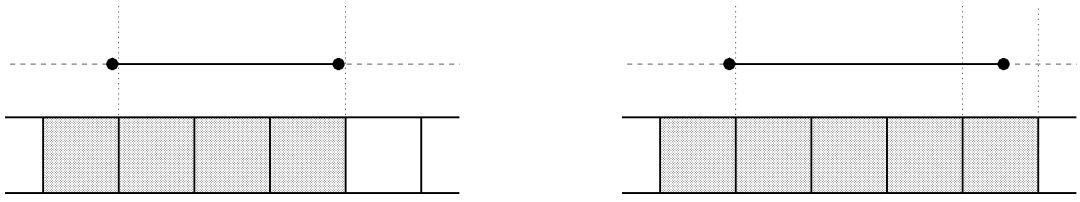


Abbildung 4.2: Es werden maximal $\lceil u \rceil + 1$ Pixel gefärbt

Es werden also in allen Fällen maximal $\lceil u \rceil + 1$ Pixel schwarz gefärbt.

Aus diesen Überlegungen ergibt sich das Intervall $\mathcal{D}(c, d)$:

Definition 4.4 Das Intervall $\mathcal{D}(c, d)$ möglicher Längen eines Dashs der Länge d unter einer Transformation w mit Kontraktionsfaktor $c \in [0, 1)$ ist folgendermaßen definiert:

$$\mathcal{D}(c, d) = [\lfloor c(d-1) \rfloor + 1, \lceil c(d-1) \rceil + 1] .$$

4.2.2 Das Intervall für die Gaps

Definition 4.5 Das Intervall $\mathcal{G}(c, g)$ beschreibt die mögliche Länge eines Gaps g unter einer diskreten Transformation w mit Kontraktionsfaktor c.

Ein Gap habe die Länge g, was auch der Anzahl seiner Pixel entspricht. Wegen der Mittelpunktrepräsentation ist der Abstand $g + 1$ vom letzten Pixelmittelpunkt des links anschließenden Dashs bis zum ersten Pixelmittelpunkt des rechts anschließenden Dashs relevant. Nach der Transformation mit w in $\mathbb{R}$ beträgt dieser Abstand $c(g + 1)$. Dieser Abstand entspricht $g + 2$ Mittelpunkten, wobei die beiden äußeren zu den betreffenden Dashs gehören.

Es stellt sich also die Frage, wieviele Pixel man maximal bzw. minimal mit dem oben beschriebenen Abstand $u := c(g + 1)$ überdecken kann.

Die untere Grenze des Intervalls

Die Anzahl der Pixel, die u überdeckt, ist dann minimal, wenn u genau an der linken Grenze eines Pixels beginnt. Das heißt, daß durch den letzten Pixelmittelpunkt des links anschließenden Dashs dieses Pixel schwarz gefärbt wird, obwohl der Dash das Pixel nicht völlig überdeckt.

Sei $r := u - \lfloor u \rfloor$. Wir unterscheiden zwei Fälle:

1. $r = 0$
   u ist ganzzahlig. Weil die Pixelbreite 1 beträgt, befindet sich der erste Pixelmittelpunkt des rechts anschließenden Dashs ebenfalls genau an der linken Grenze eines Pixels. Dieses ist nach dem Gap das erste, das schwarz gefärbt wird. Wegen $u = \lfloor u \rfloor$ werden also $\lfloor u \rfloor - 1$ Pixel weiß gelassen. Dieser Fall ist links in der Abb. 4.3 illustriert. Oben ist der

Abstand u vom letzten Pixelmittelpunkt des links anschließenden Dashs bis zum ersten Pixelmittelpunkt des rechts anschließenden Dashs zu sehen, wobei diese Pixelmittelpunkte hervorgehoben sind. Rechts und links davon sind die anschließenden Dashes angedeutet; darunter ist das Ergebnis von Δ für diesen Gap zu sehen.

2. $r > 0$
   u wird gegenüber dem obigen Fall um r verlängert. Wegen $r < 1$ und Pixelbreite $= 1$ kann der erste Pixelmittelpunkt des rechts anschließenden Dashs die nächste Pixelgrenze nie erreichen; es werden also $\lfloor u \rfloor - 1$ Pixel weiß gelassen. Ein Beispiel wird in der Abb. 4.3 rechts gezeigt.

Es werden also in allen Fällen minimal $\lfloor u \rfloor - 1$ Pixel weiß gelassen.

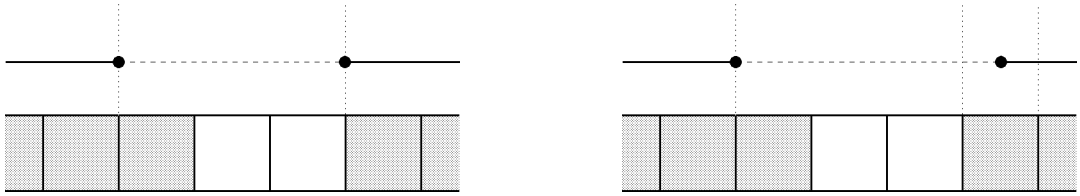


Abbildung 4.3: Es werden minimal $\lfloor u \rfloor - 1$ Pixel weiß gelassen

Die obere Grenze des Intervalls

Die Anzahl der Pixel, die u überdeckt, ist dann maximal, wenn der letzte Pixelmittelpunkt des links anschließenden Dashs möglichst weit links in einem Pixel, aber nicht auf der linken Grenze des nächsten Pixels beginnt. Das heißt, daß dieses letztere Pixel weiß gelassen wird, obwohl der Dash in dieses hineinragt.

Wir unterscheiden wiederum zwei Fälle:

1. $r = 0$
   u ist ganzzahlig. Weil die Pixelbreite 1 beträgt, befindet sich der erste Pixelmittelpunkt des rechts anschließenden Dashs an derselben Stelle in einem Pixel wie der letzte Pixelmittelpunkt des links anschließenden Dashs. Dieses ist nach dem Gap das erste, das schwarz gefärbt wird. Wegen $u = \lceil u \rceil$ werden also insgesamt $u - 1$ Pixel weiß gelassen. Dieser Fall ist links in der Abb. 4.4 illustriert.

2. $r > 0$
   u wird gegenüber dem obigen Fall um r verlängert. Wegen $r > 0$ überschreitet der erste Pixelmittelpunkt des rechts anschließenden Dashs sicher die nächste Pixelgrenze; es wird also ein Pixel mehr als im ersten Fall weiß gelassen. Das ergibt $\lceil u \rceil - 1$ schwarze Pixel. Ein Beispiel wird in der Abb. 4.4 rechts gezeigt.

Es werden also in allen Fällen maximal $\lceil u \rceil - 1$ Pixel weiß gelassen.

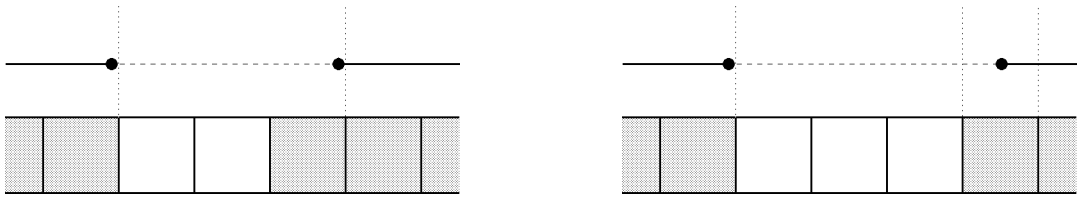


Abbildung 4.4: Es werden maximal $\lceil u \rceil - 1$ Pixel weiß gelassen

Aus diesen Überlegungen ergibt sich das Intervall $\mathcal{G}(c, g)$:

Definition 4.6 Das Intervall $\mathcal{G}(c, g)$ möglicher Längen eines Gaps der Länge g unter einer Transformation w mit Kontraktionsfaktor $c \in [0, 1)$ ist folgendermaßen definiert:

$$\mathcal{G}(c, g) = [\lfloor c(g+1) \rfloor - 1, \lceil c(g+1) \rceil - 1] \,.$$

4.2.3 Das Intervall für die Ratios

Definition 4.7 Das Intervall $\mathcal{R}(d, g, c)$ beschreibt das mögliche Verhältnis der Länge eines Dashs d zur Länge eines Gaps g unter einer diskreten Transformation w mit Kontraktionsfaktor c. $\mathcal{R}(d, g, c)$ wird *Ratio–Intervall* genannt.

Aus den oben bestimmten Intervallen kann nun das Intervall für die Ratios ermittelt werden: Seine untere Grenze wird durch das Verhältnis der minimalen Länge des Dashs zur maximalen Länge des Gaps und seine obere Grenze durch das Verhältnis der maximalen Länge des Dashs zur minimalen Länge des Gaps bestimmt. Man erhält

$$\left[\frac{\lfloor c(d-1) \rfloor + 1}{\lceil c(g+1) \rceil - 1}, \frac{\lceil c(d-1) \rceil + 1}{\max\{\lfloor c(g+1) \rfloor - 1, 1\}} \right]$$

Im Abschnitt 4.3 werden wir sehen, daß $\lceil c(g+1) \rceil - 1 \geq 1$ immer gilt. Damit $\lfloor c(g+1) \rfloor - 1 \geq 1$ gilt, wird die Maximumsfunktion verwendet.

Man kann für die untere Grenze des Intervalls eine Abschrankung ableiten, die unabhängig von c ist:

$$\begin{aligned} \lfloor c(d-1) \rfloor + 1 &> c(d-1) \\ \lceil c(g+1) \rceil - 1 &< c(g+1) \\ \frac{\lfloor c(d-1) \rfloor + 1}{\lceil c(g+1) \rceil - 1} &> \frac{c(d-1)}{c(g+1)} = \frac{d-1}{g+1} \end{aligned} \tag{4.1}$$

Wir erhalten als Intervall für die Ratios:

$$\left(\frac{d-1}{g+1}, \frac{\lceil c(d-1) \rceil + 1}{\max\{\lfloor c(g+1) \rfloor - 1, 1\}} \right]$$

4.3 Beschränkung des Kontraktionsfaktors

Damit die Ratio–Intervalle errechnet werden können, braucht man außer den Werten d und g einen Kontraktionsfaktor c.

Es wurde bereits erwähnt, daß für Kontraktionsfaktoren, die klein genug sind, Gaps geschlossen werden können. In Abhängigkeit von der Länge der Gaps, die beim Matching geschlossen werden und von derjenigen der Gaps, die offen bleiben, ergibt sich eine Beschränkung für c.

Zur genaueren Definition dieser Beschränkung werden zwei Werte definiert: *open* ist das Minimum der Längen aller offen gebliebenen Gaps, und *closed* ist das Maximum der Längen aller geschlossenen Gaps. *open* und *closed* sind aus $\mathbb{N}$. Diese Werte sind signifikant, denn ein w', das einen Gap der Länge *open* offen läßt, läßt auch alle längeren Gaps offen; ein w', das einen Gap der Länge *closed* schließt, schließt auch alle kürzeren Gaps.

Wir haben im Abschnitt 4.2.2 das Intervall $\mathcal{G}(c, g)$ ermittelt. Aus diesem lassen sich unter Verwendung von *open* und *closed* Beschränkungen für c ableiten:

1. Wir haben angenommen, daß ein Gap der Länge *open* offen bleiben kann. Dazu muß die obere Grenze von $\mathcal{G}(c, open) \geq 1$ sein:

$$\begin{aligned} \lceil c(open+1) \rceil - 1 &\geq 1 \\ \lceil c(open+1) \rceil &\geq 2 \\ c(open+1) &> 1 \\ c &> \frac{1}{open+1} \end{aligned} \tag{4.2}$$

2. Wir haben angenommen, daß ein Gap der Länge *closed* geschlossen werden kann. Dazu muß die untere Grenze von $\mathcal{G}(c, closed) \leq 0$ sein:

$$\begin{aligned} \lfloor c(closed+1) \rfloor - 1 &\leq 0 \\ \lfloor c(closed+1) \rfloor &\leq 1 \\ c(closed+1) &< 2 \\ c &< \frac{2}{closed+1} \end{aligned} \tag{4.3}$$

Es muß natürlich

$$\frac{1}{open+1} < \frac{2}{closed+1}$$

gelten, damit eine Transformation überhaupt möglich ist. Es ergibt sich wegen $open, closed \in \mathbb{N}$

$$\begin{aligned} closed+1 &< 2open+2 \\ closed &< 2open+1 \\ closed &\leq 2open. \end{aligned} \tag{4.4}$$

Es gilt also, daß es genau dann möglich ist, daß eine Transformation w Teil des gesuchten IFS ist, wenn für den Kontraktionsfaktor $c \in [0,1)$ dieses w

$$\frac{1}{open+1} < c < \frac{2}{closed+1}$$

gilt.

Wie oben erwähnt, brauchen wir c zur Berechnung des Ratio–Intervalls. Da wir für c jedoch keine nähere Beschränkung als die obige angeben können, vereinbaren wir $c_{min} := \frac{1}{open+1}$ und $c_{max} := \frac{2}{closed+1}$ und setzen entsprechend ein:

Definition 4.8 Das *Ratio–Intervall* $\mathcal{R}(d, g, open, closed)$ ist folgendermaßen definiert:

$$\mathcal{R}(d, g, open, closed) := \left(\frac{d-1}{g+1}, \frac{\lceil c_{max}(d-1) \rceil + 1}{\max\{\lfloor c_{min}(g+1) \rfloor - 1, 1\}} \right]$$

Das Ratio–Intervall ist jetzt größer als ursprünglich; diejenigen Lösungen, die dadurch hinzukommen können, werden im zweiten Teil von MatchRatio (Kapitel 5) sicher eliminiert.

4.4 Der Vorgang des Matching

Hier wird das Matching, ein wichtiger Teil von MatchRatio, beschrieben. Dazu wird die Pixelzeile P als Folge von Dashes d_i und Gaps g_i dargestellt:

$$\mathrm{P} := (d_0, g_0, d_1, g_1, \ldots, g_{m-1}, d_m).$$

Es werden die Ratios r_i des Musters P definiert:

$$r_0 := \tfrac{d_0}{g_0}, r_1 := \tfrac{d_1}{g_0}, r_2 := \tfrac{d_1}{g_1}, \ldots, r_{2(m-1)} := \tfrac{d_m}{g_{m-1}}.$$

Definition 4.9 *Matching* bedeutet das Auffinden eines Intervalls für c, sodaß ein w' mit einem Kontraktionsfaktor innerhalb dieses Intervalls das Muster P auf einen Teil G seiner selbst abbilden kann. G besteht dabei aus mindestens zwei Dashes.

Definition 4.10 Ein *Match* besteht aus dem Bereich G, auf den P abgebildet wird, aus der Anzahl der Dashes von G und aus einem Intervall für c, das durch *open* und *closed* bestimmt wird.

Hier wird der Fall eines w ohne Spiegelung erklärt. Der erste oder am weitesten links gelegene Dash von G ist durch die erste verwendete Ratio bekannt; es muß nicht der erste Dash von P sein. Der Anfangswert für *open* ist ∞, derjenige von *closed* ist 0. Der Dash, der zum ersten Ratio–Intervall beiträgt, ist immer der erste Dash von P, denn es muß ja das ganze Muster P auf G abgebildet werden.

Das Matching wird von links nach rechts durchgeführt. Dabei werden sämtliche Kombinationen von offenen und geschlossenen Gaps in P betrachtet. Es wird für jeden Gap zuerst der Fall des

offenen Gap betrachtet und die Gaps von rechts nach links geschlossen. Für jede Kombination werden die Ratio–Intervalle ermittelt und geprüft, ob die entsprechenden Ratios in den Ratio–Intervallen liegen. Je nachdem, ob Gaps beim Matching offen gelassen oder geschlossen werden, ergeben sich neue Werte für *open* und *closed*. Der Wert von *open* kann dabei nur kleiner und der von *closed* nur größer werden, sodaß sich ein immer engeres Intervall für c ergibt. Durch das Schließen von Gaps verringert sich die Anzahl von Ratio–Intervallen, sodaß deren Anzahl kleiner wird als die der Ratios. Daher kann es sein, daß zwar alle Ratio–Intervalle betrachtet werden (das muß immer der Fall sein), aber nicht alle Ratios. G endet beim Dash der letzten betrachteten Ratio und nicht unbedingt am Ende von P.

Es wird nun ein signifikanter Teil des Matching beschrieben:

Sei r_j die erste betrachtete Ratio, d_i der erste betrachtete Dash. Es wird versucht, g_i offen zu lassen. Dementsprechend wird *open* besetzt:

$$open := \min(open, g_i) \tag{4.5}$$

Hier steht g_i für die Länge von g_i. Ist jetzt $closed \leq 2open$, d.h. ist ein c noch möglich, wird das erste Ratio–Intervall geprüft:

Wenn die Aussage

$$r_j \in \mathcal{R}(d_i, g_i, open, closed)$$

zutrifft, dann bedeutet das, daß es ein w mit $c \in [c_{min}, c_{max}]$ geben kann, das bis zum aktuellen Ratio–Intervall das Muster G aus P erzeugt. Natürlich muß mit dem Matching fortgefahren werden, bis alle Ratio–Intervalle betrachtet wurden.

Es wird nun versucht, g_{i+1} offen zu lassen. Dementsprechend wird *open* besetzt: $open := \min(open, g_{i+1})$. Ist jetzt $closed \leq 2open$, d.h. ist ein c noch möglich, werden die Ratio–Intervalle geprüft:

Wenn die Aussagen

$$r_{j+1} \in \mathcal{R}(d_{i+1}, g_i, open, closed)$$

$$r_{j+2} \in \mathcal{R}(d_{i+1}, g_{i+1}, open, closed)$$

zutreffen, dann wird in der beschriebenen Weise fortgefahren.

Wenn nicht, wird g_{i+1} geschlossen. Die Dashs d_{i+1} und d_{i+2} werden dabei mit dem Gap g_{i+1} zusammengefügt, was einen großen Dash $d_{i+1,i+2}$ mit der Länge von $d_{i+1} + g_{i+1} + d_{i+2}$ ergibt. Dementsprechend wird *closed* besetzt: $closed := \max(closed, g_{i+1})$. Da nun g_{i+1} nicht mehr in *open* einfließen darf, weil dieser Gap geschlossen ist, und weil nun g_{i+2} offen bleiben soll, wird *open* entsprechend besetzt: $open := \min(open, g_{i+2})$. Dabei ist der Wert *open* in der Klammer der Wert *open* links in der Gleichung 4.5! Ist $closed \leq 2open$, werden wiederum die Ratio–Intervalle geprüft:

$$r_{j+1} \in \mathcal{R}(d_{i+1,i+2}, g_i, open, closed)$$

$$r_{j+2} \in \mathcal{R}(d_{i+1,i+2}, g_{i+2}, open, closed)$$

Sind die Ratios in den Ratio–Intervallen enthalten, wird fortgefahren. Wenn nicht, wird g_i geschlossen. Mittels Backtracking wird diese Vorgangsweise fortgesetzt, bis das letzte Ratio-Intervall erreicht ist oder wegen *open* und *closed* kein c mehr möglich ist. Wird das letzte Ratio–Intervall erfolgreich gematcht, wurde ein Match gefunden.

Ein Beispiel für einen Match ist in Abb. 4.5 zu sehen: Aus den Werten $d_0, g_0, d_1, g_1, \ldots, g_3, d_4$ (sowie *open* und *closed*) werden die Ratio–Intervalle von P berechnet. Die Ratios r_5 und r_6 werden mit den Ratio–Intervallen verglichen. Sie sind in der Abbildung eingezeichnet.

Im Laufe des Matching werden die Gaps g_0, g_2 und g_3 geschlossen. Der fertiggestellte Match überdeckt den Bereich G in der Abbildung.

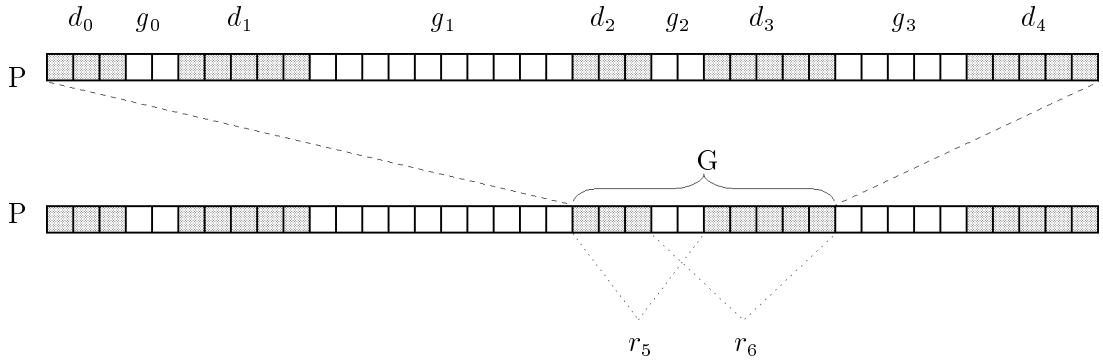


Abbildung 4.5: Ein Match der Pixelzeile P auf einen Teil G ihrer selbst

Da beim Matching immer zuerst versucht wird, die Gaps offen zu lassen, wird der Match–Algorithmus als erste Lösung die identische Abbildung mit P = G finden. Diese wird verworfen, weil sie einer Transformation entspricht, die den Kontraktionsfaktor 1 hat und deswegen nicht kontraktiv ist. Eine solche Transformation kann nicht Teil eines IFS sein.

Ziel des Match–Algorithmus ist es, den Match mit den meisten Dashes in G zu finden, da P insgesamt mit möglichst wenigen Transformationen dargestellt werden soll. Daher müssen mittels Backtracking sämtliche Kombinationen von offenen und geschlossenen Gaps getestet werden, wobei die Gaps von rechts nach links geschlossen werden.

Von den resultierenden Matches wird für den Match mit den meisten Dashes in G geprüft, ob eine Transformation existiert, die G erzeugt (Kapitel 5). Für diese Prüfung wird nur G gebraucht sowie die Information, ob der Match eine Spiegelung enthält oder nicht; das Intervall für c war nur zur Beurteilung nötig, ob der betreffende Match möglich ist. Ist die Prüfung erfolgreich, wurde eine Transformation gefunden, die Teil des gesuchten IFS ist. Ist das nicht der Fall, wird der Match mit der zweitgrößten Anzahl an Dashes in G geprüft etc.

4.5 Globale Strategie von MatchRatio

Nach diesen Vorarbeiten ist es möglich, die globale Vorgehensweise von MatchRatio zu beschreiben.

Jedes Resultat G einer durch Matching gefundenen Transformation entspricht einer Teilüberdeckung von P. Ziel von MatchRatio ist es, eine Sequenz von möglichst wenigen Teilüberdeckungen für das Muster P zu finden, sodaß die Dashes von P vollständig überdeckt sind. Dazu wird zuerst eine Sequenz von Teilüberdeckungen ohne Spiegelung und eine Sequenz von Teilüberdeckungen mit Spiegelung generiert, aus denen sich die endgültige Sequenz zusammensetzt. Diese darf aus überlappenden Teilüberdeckungen bestehen.

Zur Vereinfachung des Algorithmus werden folgende Eigenschaften für die Teilüberdeckungen festgelegt:

- Eine Teilüberdeckung kann einen oder mehrere Dashes überdecken. Falls nicht anders möglich, können auch mehrere Teilüberdeckungen einen Dash überdecken.
- Im Zwischenergebnis von MatchRatio, also innerhalb der Sequenz von Teilüberdeckungen mit Spiegelung und innerhalb der Sequenz von Teilüberdeckungen ohne Spiegelung, dürfen sich keine zwei Teilüberdeckungen überlappen.

MatchRatio beginnt mit dem ungespiegelten Fall. Beginnend bei der Ratio ganz links wird die längste Teilüberdeckung ermittelt, wie das im Abschnitt 4.4 beschrieben wurde. Existiert eine solche, wird die erste Ratio nach der ersten Teilüberdeckung als erste Ratio für den nächsten Match verwendet etc. Wird dagegen kein Match gefunden, muß der Dash, der zur ersten Ratio beiträgt, separat durch mindestens eine Transformation dargestellt werden. Dazu werden Transformationen verwendet, deren Kontraktionsfaktor so klein ist, daß sämtliche Gaps geschlossen werden. Im schlimmsten Fall muß für jedes Pixel des Dashs eine Transformation verwendet werden.

In dieser Weise wird fortgefahren, bis eine Sequenz nicht überlappender Teilüberdeckungen gefunden ist, die das gesamte Muster P überdeckt. Jede Teilüberdeckung entspricht dabei einer Transformation ohne Spiegelung.

Es folgt die Behandlung des gespiegelten Falls. Der Vorgang ist derselbe wie oben, es wird jedoch die Ratio ganz rechts als erste verwendet und von rechts nach links fortgeschritten. Das erste Ratio–Intervall ist wiederum dasjenige ganz links; auf diese Weise wird die Spiegelung realisiert. Ergebnis ist eine Sequenz nicht überlappender Teilüberdeckungen, die das gesamte Muster P überdeckt. Jede Teilüberdeckung entspricht dabei einer Transformation mit Spiegelung.

Es existieren nun zwei Überdeckungen von P; eine Sequenz von Teilüberdeckungen ohne Spiegelung und eine Sequenz von Teilüberdeckungen mit Spiegelung. Von diesen beiden Sequenzen wird eine Kombination ermittelt, sodaß P mit möglichst wenigen Teilüberdeckungen, also Transformationen, überdeckt wird.

Dazu wird ein gerichteter Graph von Teilüberdeckungen aufgebaut: Die Knoten des Graphen sind die Dashes von P sowie ein zusätzlicher Knoten, der Endknoten. Nun werden die Knoten

von links nach rechts mit Kanten versorgt: Von jedem Dash jeder Teilüberdeckung wird eine Kante zum ersten Dash der nächsten Teilüberdeckung gezogen. Bei der letzten Teilüberdeckung werden die Kanten zum Endknoten gezogen. Die Kanten sind alle mit 1 bewertet, es sei denn, ein Dash ist mit $n > 1$ Transformationen überdeckt. Dann wird eine Kante mit der Wertigkeit n zum ersten Dash der nächsten Teilüberdeckung bzw. zum Endknoten gezogen.

Nun wird der „kürzeste“ Weg von links nach rechts im Graph gesucht, sodaß die Summe der Wertigkeiten der verwendeten Kanten minimal ist. Das entspricht einer Überdeckung von P mit möglichst wenigen Teilüberdeckungen. Resultat von MatchRatio ist also eine Sequenz von möglicherweise überlappenden Teilüberdeckungen, die die Dashes von P überdecken. Die Teilüberdeckungen stehen für Transformationen mit oder ohne Spiegelung. Das gesuchte IFS, für das P ein diskreter Attraktor ist, setzt sich aus diesen Transformationen zusammen.

Ein kurzes Beispiel ist in der Abb. 4.6 gezeigt. Unter der Pixelzeile P ist die Überdeckung im ungespiegelten Fall zu sehen, darunter die Überdeckung im gespiegelten Fall.

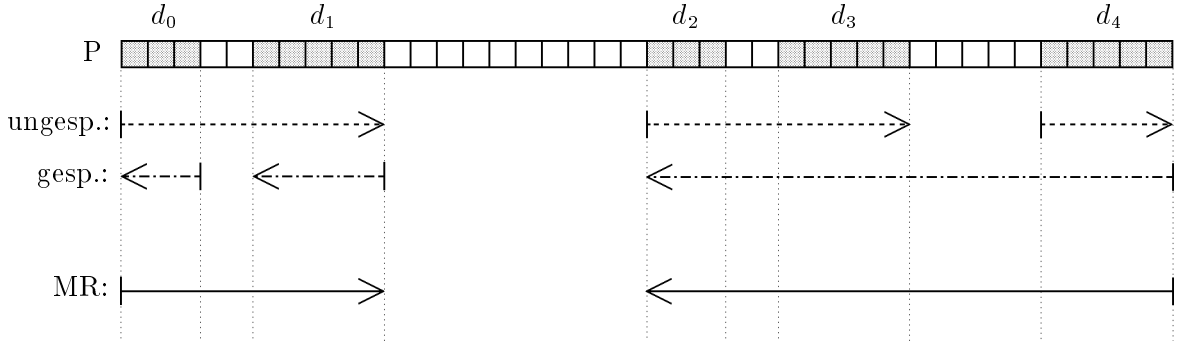


Abbildung 4.6: Die resultierende Überdeckung einer Pixelzeile P durch MatchRatio

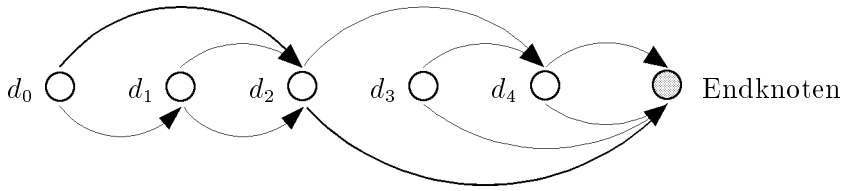


Abbildung 4.7: Der Graph zu den Überdeckungen mit und ohne Spiegelung

In der Abbildung 4.7 ist der Graph zu diesen Überdeckungen gezeigt. Die Kanten sind alle mit 1 bewertet. Der kürzeste Weg durch den Graphen ist fett gezeichnet. Er entspricht einer Überdeckung der Dashes in P mit möglichst wenigen Teilüberdeckungen. Jede Teilüberdeckung entspricht dabei einer Transformation. Diese Überdeckung, also das Resultat von MatchRatio, wird in der letzten Zeile der Abb. 4.6 gezeigt.

4.6 Zusammenfassung des Algorithmus MatchRatio

MatchRatio (Pixelzeile P)

- Behandlung des ungespiegelten Falles
  - Der Anfangsdash ist der erste Dash von links
  - **repeat**
    - **repeat**
      Es wird ein Match gesucht, also eine mögliche Transformation, die P auf einen Teil von P beginnend beim Anfangsdash abbildet.
      until alle Kombinationen von offenen und geschlossenen Gaps sind betrachtet.
    - Es wird der Match ermittelt, der die meisten Dashes erzeugt und für den eine Transformation existiert (Kapitel 5). Dieser entspricht einer Teilüberdeckung.
    - Wurde kein Match gefunden, dann wird der Anfangsdash mit mindestens einer Transformation überdeckt. Diese Transformationen entsprechen Teilüberdeckungen.
    - Der Anfangsdash ist der erste Dash rechts vom letzten überdeckten Dash.

    until der letzte Dash, also der Dash ganz rechts, ist überdeckt.
- Behandlung des gespiegelten Falles
  - Der Anfangsdash ist der erste Dash von rechts
  - **repeat**
    - **repeat**
      Es wird ein Match gesucht, also eine mögliche Transformation, die P auf einen Teil von P beginnend beim Anfangsdash abbildet.
      until alle Kombinationen von offenen und geschlossenen Gaps sind betrachtet.
    - Es wird der Match ermittelt, der die meisten Dashes erzeugt und für den eine Transformation existiert (Kapitel 5). Dieser entspricht einer Teilüberdeckung.
    - Wurde kein Match gefunden, dann wird der Anfangsdash mit mindestens einer Transformation überdeckt. Diese Transformationen entsprechen Teilüberdeckungen.
    - Der Anfangsdash ist der erste Dash links vom letzten überdeckten Dash.

    until der letzte Dash, also der Dash ganz links, ist überdeckt.
- Es existieren zwei Überdeckungen von P; eine Sequenz von Teilüberdeckungen ohne Spiegelung und eine Sequenz von Teilüberdeckungen mit Spiegelung. Ein gerichteter Graph von Teilüberdeckungen mit bewerteten Kanten wird aufgebaut (Abschnitt 4.5).
- Ein Weg von links nach rechts wird im Graph gesucht, sodaß die Summe der Wertigkeiten der verwendeten Kanten minimal ist. Das entspricht einer Überdeckung von P mit möglichst wenigen Teilüberdeckungen. Jede Teilüberdeckung entspricht einer Transformation des gesuchten IFS.

4.7 Ein Beispiel

Hier wird gezeigt, wie der Algorithmus MatchRatio ein IFS für die Pixelzeile P in der Abbildung 4.8 findet, sodaß P ein diskreter Attraktor des IFS ist. Es werden die Längen der Dashes

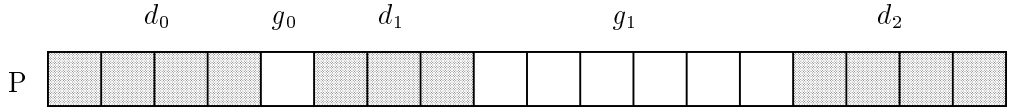


Abbildung 4.8: Für diese Pixelzeile P soll ein IFS gefunden werden

und Gaps bestimmt:

$$d_0 = 4 \qquad g_0 = 1 \qquad d_1 = 3 \qquad g_1 = 6 \qquad d_2 = 4$$

Daraus werden die Ratios errechnet:

$$r_0 = \tfrac{d_0}{g_0} = 4 \qquad r_1 = \tfrac{d_1}{g_0} = 3 \qquad r_2 = \tfrac{d_1}{g_1} = \tfrac{1}{2} \qquad r_3 = \tfrac{d_2}{g_1} = \tfrac{2}{3}$$

Zuerst werden **Transformationen ohne Spiegelung** gesucht. Der Startwert von *open* ist ∞, der Startwert von *closed* ist 0. Der **erste Match**, der gefunden wird, entspricht der identischen Abbildung. Bei diesem Match wurden sämtliche Gaps offen gelassen. Er wird aus dem im Abschnitt 4.4 angeführten Grund verworfen.

Im Zuge des Backtracking wird nun der erste Gap von rechts, g_1, geschlossen und g_0 offengelassen und ein **neuer Match** versucht. Daraus ergibt sich eine neue Folge von Dashes und Gaps:

$$d_0 = 4 \qquad g_0 = 1 \qquad d_1 = 13$$

open und *closed* erhalten ebenfalls neue Werte:

$$open = \min\{\text{Startwert}, g_0\} = 1 \qquad closed = \max\{\text{Startwert}, g_1\} = 6$$

Nun muß geprüft werden,ob $closed \leq 2open$ gilt, also ob eine solche Transformation überhaupt möglich ist.

$$6 \leq 2 \text{ ist falsch!}$$

Ein Match mit dieser Kombination aus offenen und geschlossenen Gaps kann nicht existieren, daher wird mit der nächsten Kombination fortgefahren: es wird g_0 geschlossen und g_1 offengelassen und ein **neuer Match** versucht. Daraus ergibt sich eine neue Folge von Dashes und Gaps:

$$d_0 = 8 \qquad g_0 = 6 \qquad d_1 = 4$$

open und *closed* erhalten ebenfalls neue Werte:

$$open = 6 \qquad closed = 1$$

Es muß wiederum geprüft werden, ob eine solche Transformation überhaupt möglich ist.

$$1 \leq 12 \text{ ist wahr.}$$

Zur Berechnung des Ratio–Intervalles $\mathcal{R}_0$ werden aus *open* und *closed* die Werte c_{min} und c_{max} errechnet:

$$c_{min} = \frac{1}{open+1} = \frac{1}{7} \qquad c_{max} = \frac{2}{closed+1} = 1$$

Nun kann $\mathcal{R}_0$ bestimmt werden:

$$\mathcal{R}_0(d_0, g_0, open, closed) = \left(\frac{d_0 - 1}{g_0 + 1}, \frac{\lceil c_{max}(d_0 - 1)\rceil + 1}{\max\{\lfloor c_{min}(g_0 + 1)\rfloor - 1, 1\}}\right] =$$

$$\mathcal{R}_0(8, 6, 6, 1) = \left(\frac{7}{7}, \frac{\lceil 1(7)\rceil + 1}{\max\{\lfloor \frac{1}{7}(7)\rfloor - 1, 1\}}\right] = (1, 8]$$

Es wird geprüft, ob r_0 in $\mathcal{R}_0$ enthalten ist:

$$r_0 = 4 \in \mathcal{R}_0 = (1, 8]$$

r_0 ist in $\mathcal{R}_0$ enthalten, daher wird das **nächste Ratio–Intervall** $\mathcal{R}_1$ betrachtet. Die Werte von *open* und *closed* bleiben gleich, denn es gibt keinen folgenden Gap mehr, der offen gelassen wird, und es wird auch kein weiterer Gap geschlossen. Daher bleiben auch die Werte von c_{min} und c_{max} gleich. Nun kann $\mathcal{R}_1$ aus d_1 und g_0 sowie *open* und *closed* bestimmt werden:

$$\mathcal{R}_0(4, 6, 6, 1) = \left(\frac{3}{7}, \frac{\lceil 1(3)\rceil + 1}{\max\{\lfloor \frac{1}{7}(7)\rfloor - 1, 1\}}\right] = \left(\frac{3}{7}, 4\right]$$

Es wird geprüft, ob r_1 in $\mathcal{R}_1$ enthalten ist:

$$r_1 = 3 \in \mathcal{R}_1 = \left(\frac{3}{7}, 4\right]$$

r_1 ist in $\mathcal{R}_1$ enthalten. $\mathcal{R}_1$ ist das letzte Ratio–Intervall in P, d.h. **es wurde ein Match gefunden.** Das bedeutet, daß es eine Transformation ohne Spiegelung geben kann, die P auf die Dashes d_0 und d_1 abbildet. Es muß noch geprüft werden, ob eine solche Transformation für diesen Match existiert (Kapitel 5). Die Transformation

$$w_1(x) = 0.390756x + 0.247899$$

wird gefunden, und somit wurde eine **Teilüberdeckung ohne Spiegelung für P ermittelt, die d_0 und d_1 überdeckt.**

Es wird nun versucht, einen **weiteren Match** für den letzten Dash d_2 zu finden. Da die Ratios r_0 und r_1 bereits betrachtet wurden, rücken r_2 und r_3 an ihre Stelle:

$$r_0 = \frac{1}{2} \qquad r_1 = \frac{2}{3}$$

open und *closed* werden auf ihre Startwerte zurückgesetzt; also $open = \infty$ und $closed = 0$. Da aber der Bereich G, auf den ein Match P abbildet, aus mindestens zwei Dashes bestehen muß, wird sicher kein Match gefunden. Daher wird versucht, d_2 mit möglichst wenigen Transformationen zu überdecken. Es genügt hier eine Transformation:

$$w_2(x) = 0.171429x + 14.814286$$

Es wurde also eine **Teilüberdeckung ohne Spiegelung für P gefunden, die d_2 überdeckt.**

Damit ist eine **komplette Sequenz von Teilüberdeckungen ohne Spiegelung für P gefunden.**

Nun werden **Transformationen mit Spiegelung** gesucht. Die Längen der Dashes und Gaps sind natürlich die gleichen:

$$d_0 = 4 \qquad g_0 = 1 \qquad d_1 = 3 \qquad g_1 = 6 \qquad d_2 = 4$$

Die Ratios werden in umgekehrter Reihenfolge betrachtet; so ergibt sich die Spiegelung:

$$r_0 = \tfrac{d_2}{g_1} = \tfrac{2}{3} \qquad r_1 = \tfrac{d_1}{g_1} = \tfrac{1}{2} \qquad r_2 = \tfrac{d_1}{g_0} = 3 \qquad r_3 = \tfrac{d_0}{g_0} = 4$$

open und *closed* werden auf ihre Startwerte zurückgesetzt; also $open = \infty$ und $closed = 0$. Wie oben wird beim **ersten Match** versucht, alle Gaps offen zu lassen. *open* erhält also einen neuen Wert:

$$open = 1$$

Es muß $open \leq 2closed$ gelten, was zutrifft. Daher werden c_{min} und c_{max} errechnet:

$$c_{min} = \tfrac{1}{2} \qquad c_{max} = 1$$

Nun kann $\mathcal{R}_0$ bestimmt werden:

$$\mathcal{R}_0(4,1,1,0) = \left(\frac{3}{2}, \frac{\lceil 1(3) \rceil + 1}{\max\{\lfloor \frac{1}{2}(2) \rfloor - 1, 1\}}\right] = \left(\frac{3}{2}, 7\right]$$

Es wird geprüft, ob r_0 in $\mathcal{R}_0$ enthalten ist:

$$r_0 = \frac{2}{3} \notin \mathcal{R}_0 = \left(\frac{3}{2}, 7\right]$$

r_0 ist nicht in $\mathcal{R}_0$ enthalten. Da gleich der erste Vergleich der Ratios und Ratio–Intervalle gescheitert ist, weiß man, daß ein Match mit offenem g_0 sicher nicht möglich ist. Es wird also g_0 geschlossen und g_1 offengelassen und ein **neuer Match** versucht. Daraus ergibt sich eine neue Folge von Dashes und Gaps:

$$d_0 = 8 \qquad g_0 = 6 \qquad d_1 = 4$$

open und *closed* erhalten ebenfalls neue Werte:

$$open = 6 \qquad closed = 1$$

Es muß $open \leq 2closed$ gelten, was zutrifft. Daher werden c_{min} und c_{max} errechnet:

$$c_{min} = \tfrac{1}{7} \qquad c_{max} = 1$$

Nun kann $\mathcal{R}_0$ bestimmt werden:

$$\mathcal{R}_0(8,6,6,1) = \left(\frac{7}{7}, \frac{\lceil 1(7) \rceil + 1}{\max\{\lfloor \frac{1}{7}(7) \rfloor - 1, 1\}}\right] = (1, 8]$$

Es wird geprüft, ob r_0 in $\mathcal{R}_0$ enthalten ist:

$$r_0 = \frac{2}{3} \notin \mathcal{R}_0 = (1, 8]$$

r_0 ist nicht in $\mathcal{R}_0$ enthalten; es gibt also keine Möglichkeit, für d_2 und mindestens einen Dash links davon einen Match zu finden. Daher wird versucht, d_2 mit möglichst wenigen Transformationen zu überdecken. Es genügt hier eine Transformation:

$$w_3(x) = 0.171429x + 14.814286$$

(Wenn eine Transformation nur einen Dash erzeugt, bleibt eine Spiegelung ohne Wirkung. Der Einfachheit halber wird dann in MatchRatio eine Transformation ohne Spiegelung erzeugt.) Es wurde also eine **Teilüberdeckung für P gefunden, die d_2 überdeckt.**

Nun wird ein **weiterer Match** für d_1 und mindestens einen Dash links davon gesucht. **Es wird für d_1 und d_0 ein Match gefunden**, allerdings keine Transformation. d_1 wird in der Folge mit einer Transformation dargestellt:

$$w_4(x) = 0.117647x + 5.52941$$

Das entspricht einer **Teilüberdeckung für P, die d_1 überdeckt.** Für d_0 wird kein Match gefunden; dieser Dash wird ebenfalls mit einer Transformation dargestellt:

$$w_5(x) = 0.171429x + 0.814286$$

Das entspricht einer **Teilüberdeckung für P, die d_0 überdeckt.**

Damit ist eine **komplette Sequenz von Teilüberdeckungen für P gefunden.** Wie in Abschnitt 4.5 beschrieben, wird aus den beiden gefundenen Sequenzen von Teilüberdeckungen ein **Graph** aufgebaut und der kürzeste Weg durch den Graphen gesucht. Der gefundene Weg entspricht der Sequenz der Teilüberdeckungen ohne Spiegelung.

Das von MatchRatio gefundene IFS besteht also aus zwei Transformationen und ist das folgende:

$$IFS = \{w_1(x) = 0.390756x + 0.247899, w_2(x) = 0.171429x + 14.814286\}$$

4.8 Eigenschaften von MatchRatio

Das Verhalten des Algorithmus MatchRatio wird in diesem Abschnitt genauer betrachtet. In der folgenden Aufzählung werden verschiedene Fragen aufgeworfen und mittels Experimenten geklärt. Dazu wurde der folgende Test gemacht: MatchRatio wurde auf einem sequentiellen Rechner implementiert und 4700 mal auf ebensoviele verschiedene Pixelzeilen angewendet. Die Pixelzeilen waren je 1000 Pixel lang und wurden durch zufällige IFS generiert, um fraktale Muster zu erhalten. Die Anwendung von MatchRatio ist aufgrund der Struktur des Algorithmus hauptsächlich bei fraktalen Mustern sinnvoll; deshalb wurde dieser Test gewählt. MatchRatio benötigte durchschnittlich 5.99 Sekunden, um für eine Pixelzeile ein IFS zu finden.

- Beim Matching im Abschnitt 4.4 werden alle möglichen, als erste jedoch diejenigen Matches mit möglichst wenigen geschlossenen Gaps ermittelt. **Daher sollte man annehmen, daß diejenige Teilüberdeckung, die die meisten Dashes überdeckt, immer**

als erste (nach der identischen Transformation) gefunden wird. Wenn dem so wäre, müßte man nicht alle möglichen Matches bestimmen, denn es wird ja gerade diejenige Teilüberdeckung weiterverwendet, die die meisten Dashes überdeckt. **Das oben beschriebene Experiment hat jedoch gezeigt, daß in 75% aller Fälle diejenige Teilüberdeckung mit den meisten Dashes als erste gefunden wurde und in 25% der Fälle als zweite. Es ist also unumgänglich, alle möglichen Matches zu bestimmen.**

Der Fall, daß diejenige Teilüberdeckung, die die meisten Dashes überdeckt, nicht als erste gefunden wird, ist in der Abb. 4.9 skizziert. Die Pixelzeile P ist als Folge von Dashes (die fetten Linien in der Abbildung) und Gaps dargestellt. Wir betrachten den ungespiegelten Fall. Oberhalb und unterhalb von P ist jeweils ein Match dargestellt: Die Bögen symbolisieren Dashes, die Gaps von P innerhalb eines Bogens sind also im Match geschlossen worden. Wir nehmen an, daß für alle dargestellten Matches von P Transformationen existieren. Der obere Match wird aufgrund der Struktur des Backtracking zuerst gefunden, wobei die Gaps g_2 und g_3 geschlossen werden; er ergibt drei Dashes. Der untere Match wird später durch das Schließen des Gaps g_0 gefunden, ergibt aber vier Dashes.

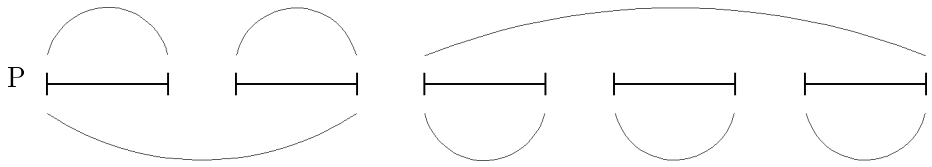


Abbildung 4.9: Zwei mögliche Überdeckungen von P

- Sei P eine Pixelzeile, die durch ein IFS mit n Transformationen erzeugt wurde. Die Ergebnisse der einzelnen Transformationen überlappen sich nicht. **Es stellt sich die Frage, wie viele Transformationen im Vergleich dazu das von MatchRatio für P gefundene IFS hat.** Die Ergebnisse des oben beschriebenen Experiments sind in der Tabelle 4.1 dargestellt. In der oberen Zeile ist die Differenz der Anzahl der Transformationen des IFS, das P erzeugt hat, und der Anzahl der Transformationen des IFS, das MatchRatio für P gefunden hat, eingetragen. In der zweiten Zeile steht, wie oft der obenstehende Fall im Experiment aufgetreten ist.

  Es zeigt sich, daß MatchRatio meistens ein IFS für P findet, das gleich viele Transformationen hat wie das IFS, mit dem P erzeugt wurde. Am zweithäufigsten fand MatchRatio ein IFS mit einer Transformation weniger als das Ausgangs–IFS. MatchRatio fand IFS mit bis zu vier Transformationen weniger als die betreffenden Ausgangs–IFS, allerdings auch IFS mit bis zu acht Transformationen mehr. Letzteres trat allerdings im Experiment nur einmal auf.

  In der Abb. 4.10 wird gezeigt, wie es zustande kommt, daß MatchRatio ein IFS mit mehr Transformationen als das Ausgangs–IFS findet: P wird durch die zugehörige Folge von

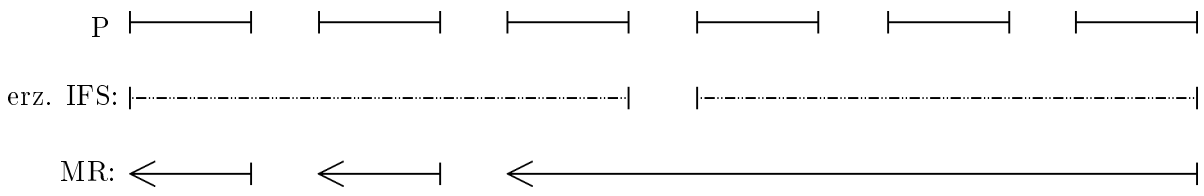


Abbildung 4.10: Ein suboptimales Ergebnis von MatchRatio

Dashes und Gaps dargestellt. Das gezeigte P wurde durch ein IFS mit 2 Transformationen mit Spiegelung erzeugt; diese sind unter P dargestellt. Darunter ist das IFS zu sehen, das von MatchRatio gefunden wurde: MatchRatio fand bei der Suche nach Transformationen ohne Spiegelung eine schlechte Überdeckung, bei der Suche nach Transformationen mit Spiegelung dagegen eine Teilüberdeckung, die mehr Dashes überdeckt als die erzeugende Transformation (die Teilüberdeckung ganz rechts in der Abbildung). Diese Teilüberdeckung wurde beibehalten, weil sie die meisten Dashes überdeckt. Das hatte zur Folge, daß für die restlichen beiden Dashes keine gute Teilüberdeckung gefunden wurde: jeder Dash mußte durch eine separate Teilüberdeckung dargestellt werden. Das ergab eine Überdeckung mit 3 Transformationen mit Spiegelung, die MatchRatio als beste Überdeckung fand. Das resultierende IFS hat also eine Transformation mehr als das IFS, das die Pixelzeile P erzeugt hat.

Eine Möglichkeit, solche Fälle zu verhindern, wäre, in MatchRatio beginnend mit *jedem* Dash von P eine möglichst lange Teilüberdeckung zu suchen und dann die Überdeckung mit möglichst wenigen Teilüberdeckungen zu ermitteln. Das bedeutet natürlich einen höheren Aufwand. Die Ergebnisse in der Tabelle 4.1 haben gezeigt, daß solche Fälle nicht oft auftreten; eine Erweiterung von MatchRatio scheint also nicht sinnvoll.

- Wir haben wiederholt festgestellt, daß MatchRatio für eine binäre Pixelzeile P ein IFS findet, für das P ein diskreter Attraktor ist. Im Kapitel 3 wurde jedoch erwähnt, daß der diskrete Attraktor eines IFS nicht eindeutig ist. **Gibt man einen diskreten Attraktor des von MatchRatio für P gefundenen IFS aus, so kann es theoretisch sein, daß ein anderer diskreter Attraktor des IFS dargestellt wird als die Pixelzeile P.**

  Im obigen Experiment wurde ein diskreter Attraktor von jedem gefundenen IFS ausgeben. **In 99.7% der Fälle war der diskrete Attraktor mit P ident.** In den anderen Fällen war das Ergebnis P sehr ähnlich, denn die Differenz zwischen den verschiedenen diskreten Attraktoren eines IFS ist beschränkt:

# Tr.d.erz.IFS-# Tr.d.gef.IFS	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
# Fälle	2	27	148	845	3449	165	40	17	2	1	2	1	1

Tabelle 4.1: Statistik der Anzahl der Transformationen von IFS

Im Kapitel 3 wurde festgehalten, daß die Differenz $d_h(\mathcal{A}, \mathcal{A}_d)$ zwischen dem abgetasteten Attraktor $\mathcal{A}$ und jedem diskreten Attraktor $\mathcal{A}_d$ eines IFS beschränkt ist. Zwei diskrete Attraktoren eines IFS können demnach nicht mehr als $2d_h(\mathcal{A}, \mathcal{A}_d)$ voneinander entfernt sein. Wir erhalten als Differenz zweier diskreter Attraktoren $\mathcal{A}_{d1}, \mathcal{A}_{d2}$ eines IFS mit Kontraktionsfaktor c

$$d_h(\mathcal{A}_{d1}, \mathcal{A}_{d2}) \leq \frac{1}{1-c} \tag{4.6}$$

- Wie beschrieben, wird jede Kombination von offenen und geschlossenen Gaps in P auf einen Match geprüft. **Das würde bei m Gaps in P einen Aufwand von 2^m geprüften Kombinationen ergeben. In der Praxis wurden jedoch viel weniger Kombinationen von offenen und geschlossenen Gaps betrachtet.** Beim obigen Experiment wurde für P mit verschieden vielen Gaps gezählt, wie oft eine Kombination von offenen und geschlossenen Gaps betrachtet wurde. Im Diagramm der Abbildung 4.11 ist die Anzahl der Gaps in P in x–Richtung aufgetragen, die Anzahl der betrachteten Kombinationen in y–Richtung. Der Aufwand ist deshalb um einiges geringer als 2^m bei m Gaps, weil durch

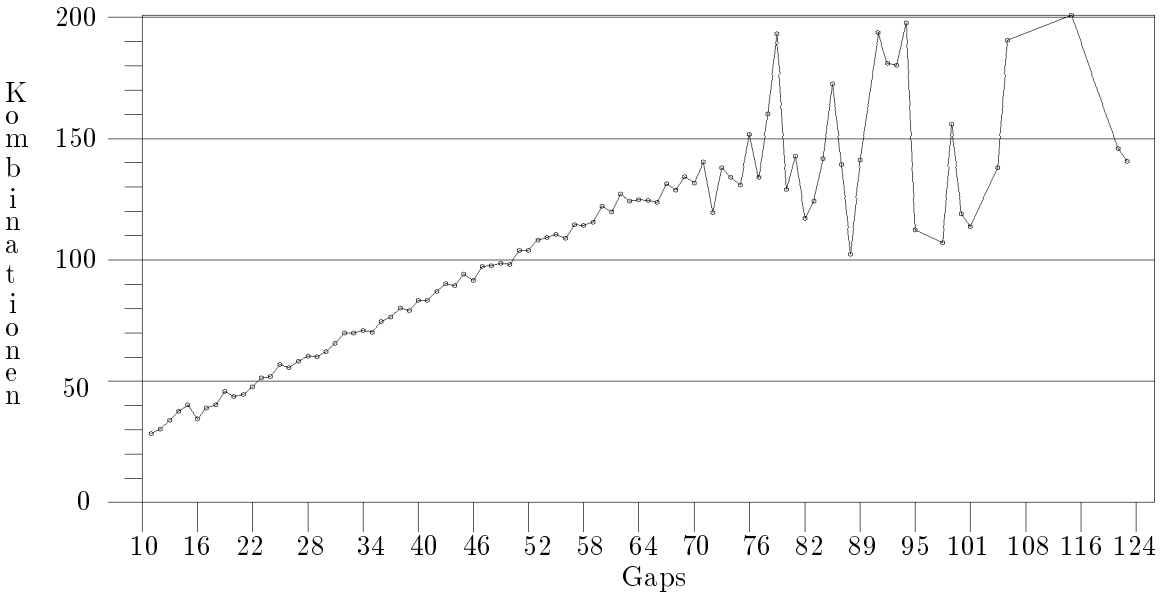


Abbildung 4.11: Anzahl der betrachteten Kombinationen für P mit verschieden vielen Gaps

ein Scheitern im Matching viele Kombinationen gar nicht erst betrachtet werden.

Kapitel 5

MatchRatio – Ermittlung von Transformationen

In diesem Kapitel werden einige neue Begriffe eingeführt, die bei ihrem ersten Auftreten kursiv geschrieben sind. Ihre Bedeutung kann im Abschnitt 5.5 nachgeschlagen werden.

5.1 Eine zur gesuchten Transformation äquivalente Lösungsgerade

Nachdem durch den Algorithmus MatchRatio eine mögliche Transformation für einen Teil des diskreten Attraktors gefunden wurde, muß nun eine Transformation w' gefunden werden, die den diskreten Attraktor auf diesen Teil abbildet. Der betreffende Teil des diskreten Attraktors ist durch MatchRatio bestimmt und heißt Zielpixelmuster G, der diskrete Attraktor selbst ist die Pixelzeile P.

5.1.1 Vorbereitungen

Ein Pixelmuster G wurde ermittelt, das Teil der Pixelzeile P ist. G ist dabei kürzer als P, und das erste und das letzte Pixel von G sind schwarz. Die Pixel von P werden mit Indizes versehen, wodurch sich gleichzeitig Indizes für die Pixel in G ergeben. Der Index eines Pixels ist außerdem die Koordinate des am weitesten links gelegenen Punktes dieses Pixels. Die Pixelzeile P wird als Folge von schwarzen und weißen Zusammenhangskomponenten betrachtet, wobei die schwarzen Zusammenhangskomponenten wie in Kapitel 4 als Dashes bezeichnet werden.

Wir suchen nun eine Transformation $w = cx + d$, die das Muster G liefert, wenn sie diskret auf P angewendet wird:

$$\mathrm{G} = w'(\mathrm{P}) \tag{5.1}$$

Dabei werden die Mittelpunkte der Pixel von P transformiert, wie das im Abschnitt 3.2 beschrieben wurde. Die Koordinate des Mittelpunktes eines Pixels ergibt sich aus dessen Index $+0.5$, die Koordinate des transformierten Mittelpunktes lautet $\lfloor w(\text{Index}+0.5)\rfloor$. Wird ein Pixel in G von einem transformierten Pixelmittelpunkt getroffen, dann und nur dann wird es unter w' schwarz gefärbt werden. Werden genau die schwarzen Pixel von G unter w' schwarz gefärbt, dann gilt $\mathrm{G} = w'(\mathrm{P})$.

Satz 5.1 *Zur Bestimmung der Transformation w' genügt es, die Anfangs– und Endpixel aller Dashes zu betrachten.*

Beweis: w muß eine kontraktive Transformation sein. Das bedeutet, daß der Abstand zwischen den Pixelmittelpunkten nach Anwendung von w jedenfalls kleiner ist als vor Anwendung von w. Wird also ein Dash transformiert, dann wird jedes Pixel in G zwischen dem transformierten Anfangs– und dem transformierten Endpixel des Dashs von einem transformierten Mittelpunkt getroffen werden. Der transformierte Dash bleibt eine Zusammenhangskomponente; kein weißes Pixel kann entstehen. □

Das Problem kann graphisch recht gut veranschaulicht werden (Abb. 5.1). Die fetten waagrech-

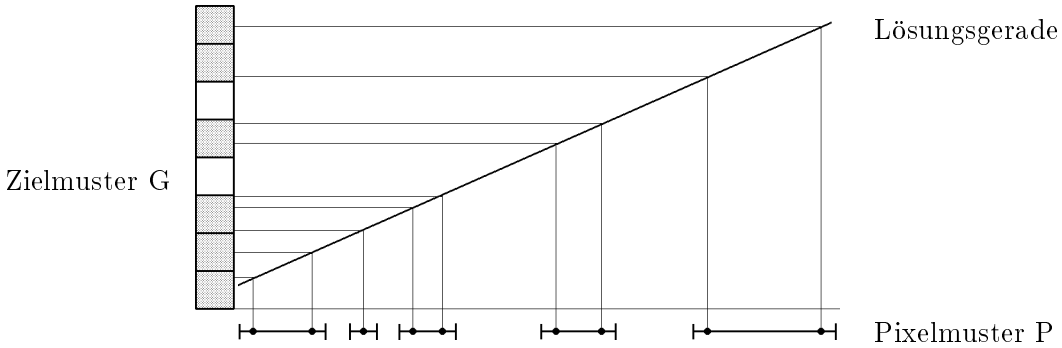


Abbildung 5.1: Graphische Veranschaulichung des Problems im ungespiegelten Fall

ten Linien in Abb. 5.1 sind die fünf Dashes von P. Das Pixelmuster G ist links von unten nach oben aufgetragen. Die senkrechten Linien durch die Mittelpunkte der Anfangs– und Endpixel der Dashes bezeichnen wir als *vlines*. Eine vline durch das Anfangspixel eines Dashs heißt *begin–vline*, eine vline durch das Endpixel eines Dashs *end–vline*. Die schräge Gerade ist die *Lösungsgerade* und stellt die gesuchte Transformation w dar. Sie ist so gewählt, daß die Mittelpunkte der Anfangs– und Endpixel der Dashes von P auf die Anfangs– und Endpixel der Dashes in G transformiert werden.

Dies wird graphisch veranschaulicht, indem die Schnittpunkte der vlines mit der Lösungsgerade, die für w steht, auf G projiziert werden. Die Pixel in G, die von den projizierten Schnittpunkten getroffen werden, sind schwarz. Weiters sind für jeden Dash aus den oben genannten Gründen

alle Pixel zwischen dem projizierten Schnittpunkt der begin–vline mit der Lösungsgerade und dem projizierten Schnittpunkt der end–vline mit der Lösungsgerade schwarz.

Abb. 5.1 zeigt eine Lösungsgerade mit positiver Steigung, was einer Transformation entspricht, die keine Spiegelung enthält. Abb. 5.2 zeigt die graphische Veranschaulichung des Problems im gespiegelten Fall. Ob eine Transformation mit oder ohne Spiegelung gefunden werden soll, wurde durch MatchRatio bereits bestimmt.

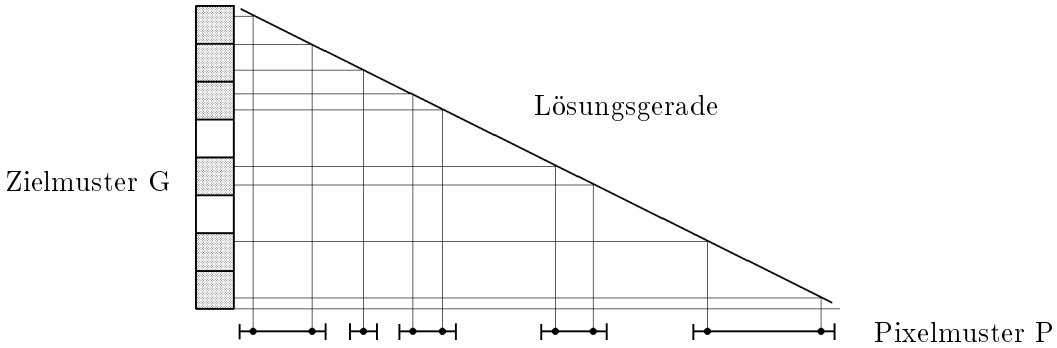


Abbildung 5.2: Graphische Veranschaulichung des Problems im gespiegelten Fall

Zur weiteren Behandlung des Problems werden wir die graphische Darstellung von Abb. 5.1 bzw. Abb. 5.2 benutzen, da diese sehr anschaulich ist. Das ist deshalb möglich, weil die zur gesuchten Transformation w' gehörige Transformation $w = cx + d$ einer Geradengleichung entspricht; w' kann sofort aus w ermittelt werden. Somit ist das Problem äqivalent zur Suche nach einer Lösungsgerade. Zu diesem Zweck denke man sich in Abb. 5.1 ein Koordinatensystem, das seinen Ursprung im Kreuzungspunkt der Pixelmuster P und G hat. Die Pixel in G behalten jedoch die Indizes, die sie innerhalb des Musters P innehatten. Das Koordinatensystem orientiert sich in x–Richtung an den Indizes von P und in y–Richtung an den Indizes von G. Zwischen diesen ganzzahligen Werten existieren reelle Werte.

5.1.2 Die Beschränkungsgeraden upper und lower

Der Bereich, in dem die gesuchte Transformation liegen darf, kann eingeschränkt werden, denn w' muß zwei Bedingungen erfüllen:

1. Das erste Pixel von P muß auf das erste Pixel von G abgebildet werden.
2. Das letzte Pixel von P muß auf das letzte Pixel von G abgebildet werden.

Das heißt genauer ausgedrückt, daß der Mittelpunkt des ersten Pixels von P unter w in den Bereich des ersten Pixels von G transformiert werden muß. Ebenso muß der Mittelpunkt des letzten Pixels von P unter w in den Bereich des letzten Pixels von G transformiert werden.

Die Abb. 5.3 illustriert die Beschränkung im ungespiegelten Fall: Der linke der mit dem Symbol

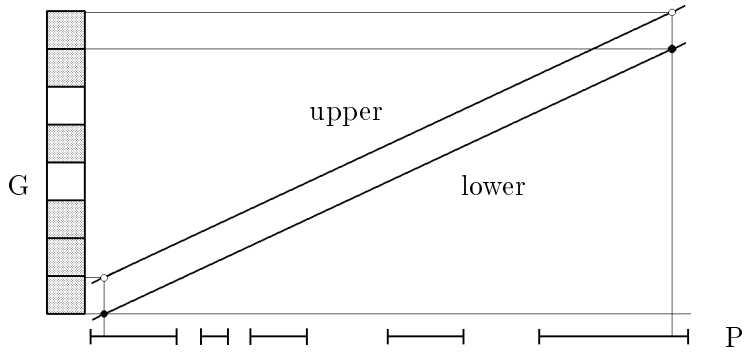


Abbildung 5.3: Die Beschränkungsgeraden upper und lower

• gekennzeichneten Punkte ist der kleinste Wert, den der Mittelpunkt des ersten Pixels von P unter w annehmen darf, damit das erste Pixel von P auf das erste Pixel von G abgebildet wird. Der rechte der mit dem Symbol • gekennzeichneten Punkte ist der kleinste Wert, den der Mittelpunkt des letzten Pixels von P unter w annehmen darf, damit das letzte Pixel von P auf das letzte Pixel von G abgebildet wird.

Anhand dieser beiden Werte ist eine Gerade definiert.

Definition 5.2 Die Gerade *lower* stellt eine untere Beschränkung für die Lösungsgerade und damit für w dar. Sie garantiert zusammen mit der Gerade *upper* (Def. 5.3), daß die beiden oben erwähnten Bedingungen eingehalten werden.

Ähnlich wie oben ist nun der linke der mit dem Symbol ∘ gekennzeichneten Punkte der größte Wert, den der Mittelpunkt des ersten Pixels von P unter w annehmen darf, damit das erste Pixel von P auf das erste Pixel von G abgebildet wird. Der rechte der mit dem Symbol ∘ gekennzeichneten Punkte ist der größte Wert, den der Mittelpunkt des letzten Pixels von P unter w annehmen darf, damit das letzte Pixel von P auf das letzte Pixel von G abgebildet wird.

Diese beiden Werte definieren eine zweite Gerade.

Definition 5.3 Die Gerade *upper* stellt eine obere Beschränkung für die Lösungsgerade und damit für w dar. Sie garantiert zusammen mit der Gerade *lower* (Def. 5.2), daß die beiden oben erwähnten Bedingungen eingehalten werden.

Behandlung von numerischen Ungenauigkeiten

Der größte Wert, den der Mittelpunkt eines Pixels in P annehmen darf, um in ein bestimmtes Pixel in G abgebildet zu werden, wird durch die Maschinenarithmetik beeinflußt. Er muß daher

genau definiert werden.

Ein Pixel in G mit dem Index i wird mit dem Intervall $[i, i+1)$ identifiziert. Ab $i+1$ beginnt das Pixel mit dem Index $i+1$. Werden nun in einer Maschinenarithmetik annähernd gleich große Werte voneinander subtrahiert, führt das zu numerischen Ungenauigkeiten. Wegen dieser numerischen Ungenauigkeiten könnte z.B. der Mittelpunkt eines Pixels in P gerade auf oder knapp über den Wert $i+1$ fallen, obwohl er bei exakter Berechnung auf das Pixel i transformiert würde. Deshalb ist es notwendig, sicherzustellen, daß solche numerischen Ungenauigkeiten keine Fehler im Algorithmus erzeugen. Es wird also ein Bereich der Breite ϵ am oberen Ende eines jeden Pixelbereichs definiert. Lösungen, die einen Pixelmittelpunkt in einen Bereich ϵ transformieren, werden nicht zugelassen. ϵ muß so groß sein, daß sein Wert in der Fließkommaarithmetik des betreffenden Rechners berücksichtigt wird, d.h. für gültige Indexwerte x muß $x + \epsilon > x$ gelten.

Abb. 5.4 zeigt diese Maßnahme für das erste Pixel aus G im ungespiegelten Fall.

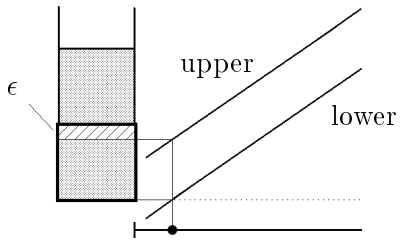


Abbildung 5.4: Der Bereich ϵ

5.1.3 Die vlines

Innerhalb der Beschränkung von upper und lower müssen weitere Beschränkungen definiert werden, die sich durch das Pixelmuster G ergeben.

Dazu werden die erwähnten vlines verwendet. Wie im Abschnitt 5.1.1 erklärt, müssen nur die Anfangs- und Endpixel der Dashes betrachtet werden. Diesen entsprechen die begin– bzw. end–vlines. Schneidet man eine vline mit den Beschränkungsgeraden upper und lower, so erhält man die folgenden Werte:

Definition 5.4 Ein *top–Punkt* einer vline ist der Maximalwert, den der Mittelpunkt des Pixels des zur betreffenden vline gehörenden Dashs unter allen w annehmen kann, die upper und lower erfüllen. (x_{top},y_{top}) sind die Koordinaten des top–Punktes.

Definition 5.5 Ein *bottom–Punkt* einer vline ist der Minimalwert, den der Mittelpunkt des Pixels des zur betreffenden vline gehörenden Dashs unter allen w annehmen kann, die upper und lower erfüllen. Die Koordinaten des bottom–Punktes sind (x_{bottom},y_{bottom}).

Für jede vline werden der top–Punkt und der bottom–Punkt ermittelt. Projiziert man den top– und den bottom–Punkt auf G, so kann man die Indizes derjenigen Pixel in G feststellen, in die der top– und der bottom–Punkt fallen. Für jede vline werden diese Indizes *top* und *bottom* zur späteren Verwendung festgehalten. Sie bezeichnen zugleich die Koordinate des in den Abbildungen am weitesten unten gelegenen Punktes des Pixels. Ein Beispiel ist in der Abb. 5.5 dargestellt. Es wird darauf hingewiesen, daß diese und die folgenden Abbildungen Skizzen sind, auf denen nur das jeweils Wesentliche exakt dargestellt wird.

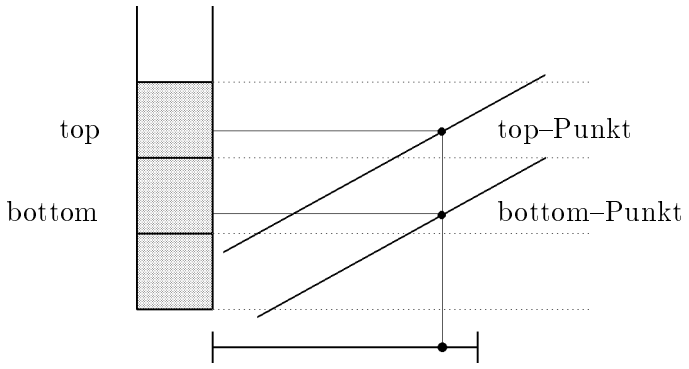


Abbildung 5.5: Graphische Veranschaulichung der Werte top–Punkt, bottom–Punkt, top und bottom

5.1.4 Die constraints

Die Intervalle $[y_{bottom}, y_{top}]$ der vlines bezeichnen den durch upper und lower beschränkten Wertebereich der Schnittpunkte von w mit den vlines. Der Wertebereich der Schnittpunkte von w mit den vlines wird im folgenden vereinfachend als Wertebereich bezeichnet werden. Eine zusätzliche Einschränkung dieses Wertebereichs ergibt sich durch die Tatsache, daß unter w' bestimmte Pixel im Zielmuster G schwarz und die anderen weiß sein müssen.

Betrachten wir das Beispiel einer gespiegelten Transformation in Abb. 5.6.

Wir gehen vom Zielpixel x in G aus. Dieses Pixel muß unter w' weiß bleiben. Es existiert jedoch eine vline, deren zugehöriger Pixelmittelpunkt in P unter den Beschränkungen von upper und lower auf x abgebildet werden kann. Deshalb wird der Wertebereich weiter beschränkt; er reicht jetzt von y_{bottom} bis top - ϵ. Der Wert ϵ ist im Abschnitt 5.1.2 näher erklärt. Wenn nun die zu w gehörende Lösungsgerade die vline innerhalb dieses eingeschränkten Wertebereichs schneidet, kann die gesuchte Transformation w' den Mittelpunkt des Pixels dieser vline keinesfalls auf x abbilden. Dann bleibt x weiß.

Definition 5.6 Ein eingeschränkter Wertebereich auf einer vline, der es ermöglicht, daß ein Pixel in G unter w' die korrekte Farbe erhält, wird ein *constraint* für dieses Pixel genannt.

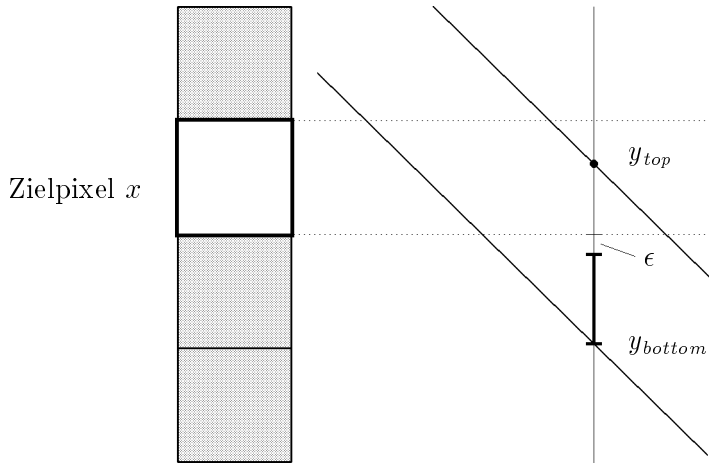


Abbildung 5.6: Ein constraint im gespiegelten Fall

Dabei unterscheidet man zwei Arten von constraints:

- *floor–constraints* oder *floors*
  Ihr Wertebereich beginnt beim bottom–Punkt der betreffenden vline; man könnte sagen, sie „kommen vom Boden“.
- *ceiling–constraints* oder *ceilings*
  Ihr Wertebereich beginnt beim top–Punkt der betreffenden vline; man könnte sagen, sie „kommen von der Decke“.

Das andere Ende dieser Wertebereiche kennzeichnet ein sogenannter *floor–Punkt* bzw. *ceiling–Punkt*. Auf diese Werte wird im Abschnitt 5.1.5 genauer eingegangen.

Die Abb. 5.7 und 5.8 zeigen jeweils den ungespiegelten Fall. Links ist der constraint dargestellt, rechts die entsprechende Kurzform, die in späteren Abbildungen verwendet werden wird. Das Symbol $\times$ bezeichnet den floor–Punkt, das Symbol $\bullet$ den ceiling–Punkt. In der Abb. 5.6, die zur Erklärung der constraints verwendet wurde, finden wir ein Beispiel für einen floor im gespiegelten Fall.

5.1.5 Relevante vlines

Nicht alle vlines sind zur Bestimmung der Lösungsgerade relevant. Welche vlines relevant sind, hängt von der Lage der Zielpixel in G ab. Es gibt zwei Fälle. Jeweils zwei vlines sind für ein Zielpixel x relevant: In den Abb. 5.9 und 5.10 werden sie mit L für *left–vline* und R für *right–vline* bezeichnet.

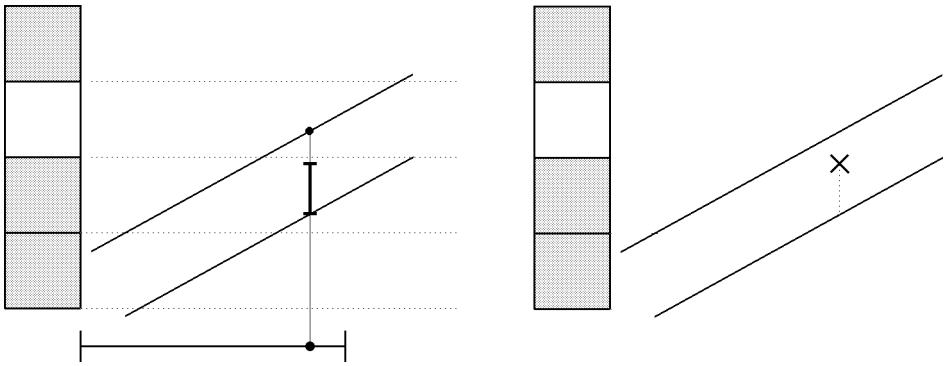


Abbildung 5.7: Ein floor-constraint und die entsprechende Kurzform

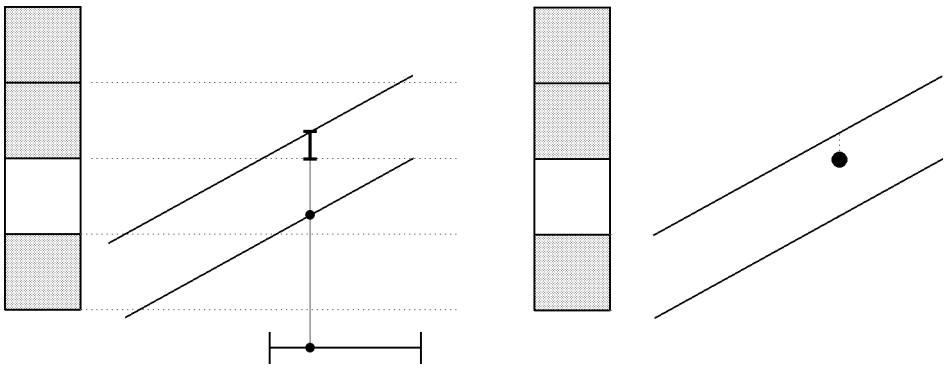


Abbildung 5.8: Ein ceiling-constraint und die entsprechende Kurzform

Definition 5.7 Für schwarze und weiße Zielpixel im ungespiegelten Fall (Abb. 5.9) ist die *left–vline* die am weitesten rechts liegende end–vline, deren top in x liegt. Die *right–vline* ist die am weitesten links liegende begin–vline, deren bottom in x liegt.

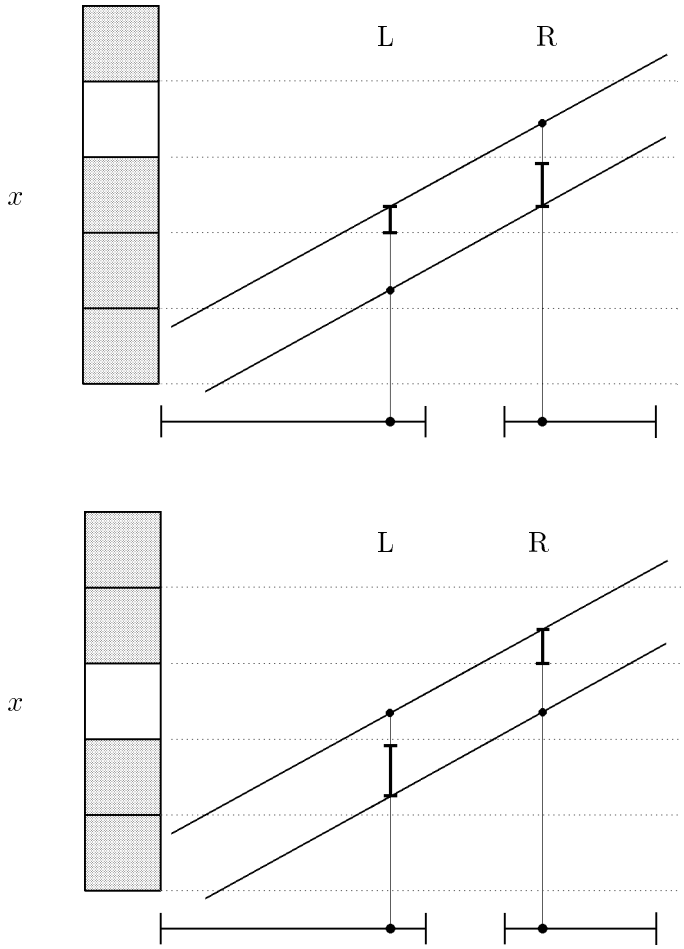


Abbildung 5.9: Schwarzes (oben) und weißes Zielpixel x (unten) im ungespiegelten Fall

Definition 5.8 Für schwarze und weiße Zielpixel im gespiegelten Fall (Abb. 5.10) ist die *left–vline* die am weitesten rechts liegende end–vline, deren bottom in x liegt. Die *right–vline* ist die am weitesten links liegende begin–vline, deren top in x liegt.

Die Farbe des jeweiligen Zielpixels bestimmt die Wertebereiche der left–vlines und right–vlines. Sie werden wie in Abschnitt 5.1.4 gesetzt, wie man in den Abb. 5.9 und 5.10 nachvollziehen kann.

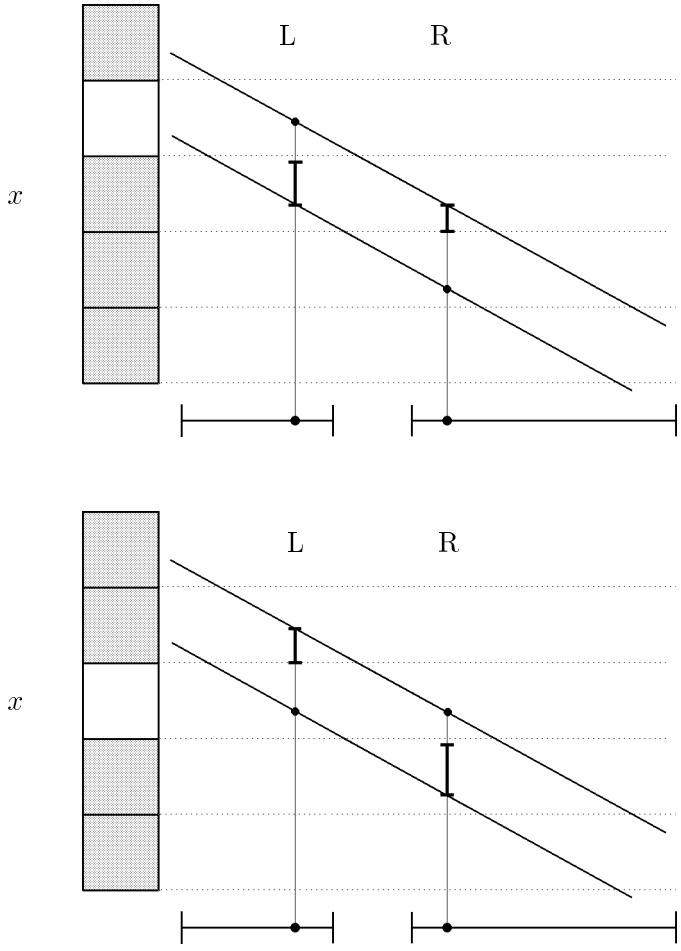


Abbildung 5.10: Schwarzes (oben) und weißes Zielpixel x (unten) im gespiegelten Fall

ungespiegelter Fall	gespiegelter Fall	(Farbe d. Zielpixels)
$y_{ceil} = \text{bottom} + 1$	$y_{floor} = \text{top} - \epsilon$	(weiß)
$y_{floor} = \text{bottom} + 1 - \epsilon$	$y_{ceil} = \text{top}$	(schwarz)

Tabelle 5.1: Die Werte y_{floor} und y_{ceil} für begin–vlines

ungespiegelter Fall	gespiegelter Fall	(Farbe d. Zielpixels)
$y_{floor} = \text{top} - \epsilon$	$y_{ceil} = \text{bottom} + 1$	(weiß)
$y_{ceil} = \text{top}$	$y_{floor} = \text{bottom} + 1 - \epsilon$	(schwarz)

Tabelle 5.2: Die Werte y_{floor} und y_{ceil} für end–vlines

Schneidet die Lösungsgerade die gerade bestimmten Wertebereiche von L und R eines Zielpixels x, so erhält x unter w' die gewünschte Farbe. Bei den schwarzen Zielpixeln muß die Lösungsgerade die Wertebereiche von L oder R schneiden, bei den weißen immer die Wertebereiche von L und R. Es ist jedoch durchaus möglich, daß L oder R für ein Zielpixel nicht existieren, weil es einfach keine vline an der betreffenden Stelle gibt.

Aus diesen constraint–Möglichkeiten ergeben sich die y–Koordinate y_{floor} der floor–Punkte und die y–Koordinate y_{ceil} der ceiling–Punkte. (Die x–Koordinaten x_{floor} bzw. x_{ceil} entsprechen der x–Koordinate der zugehörigen vline.) Grundsätzlich ist für jede vline ein floor–Punkt und ein ceiling–Punkt definiert; nicht alle diese Werte werden jedoch weiterverwendet. Zur besseren Anschaulichkeit ist in der letzten Spalte der beiden folgenden Tabellen das Zielpixel aufgeführt, für das der betreffende floor–Punkt oder ceiling–Punkt i.a. verwendet werden wird.

Die Tabelle 5.1 enthält die Werte für die begin–vlines. In der Abb. 5.11 oben sind Beispiele für die floor– und ceiling–Punkte der ersten Reihe in der Tabelle 5.1 dargestellt. Es werden links ein ceiling–constraint für ein weißes Zielpixel und rechts ein floor–constraint für ein weißes Zielpixel gezeigt. In der Abb. 5.11 unten sind Beispiele für die floor– und ceiling–Punkte der zweiten Reihe in der Tabelle 5.1 dargestellt. Links ist ein floor–constraint für ein schwarzes Zielpixel und rechts ein ceiling–constraint für ein schwarzes Zielpixel gezeigt.

Die Tabelle 5.2 enthält die Werte für die end–vlines. In der Abb. 5.12 oben sind Beispiele für die floor– und ceiling–Punkte der ersten Reihe in der Tabelle 5.2 dargestellt. Es werden links ein floor–constraint für ein weißes Zielpixel und rechts ein ceiling–constraint für ein weißes Zielpixel gezeigt. In der Abb. 5.12 unten sind Beispiele für die floor– und ceiling–Punkte der zweiten Reihe in der Tabelle 5.2 dargestellt. Links ist ein ceiling–constraint für ein schwarzes Zielpixel und rechts ein floor–constraint für ein schwarzes Zielpixel gezeigt.

Die vlines des ersten und des letzten Pixels von P stellen einen Sonderfall dar. Diese Pixel sind jedenfalls schwarz; dadurch wurden die Beschränkungsgeraden upper und lower festgelegt. Um die Beschränkungsgeraden einzuhalten, werden diese beiden vlines gemäß Abb. 5.13 mit floor–Punkten (dargestellt durch das Symbol $\times$) und ceiling–Punkten (dargestellt durch das

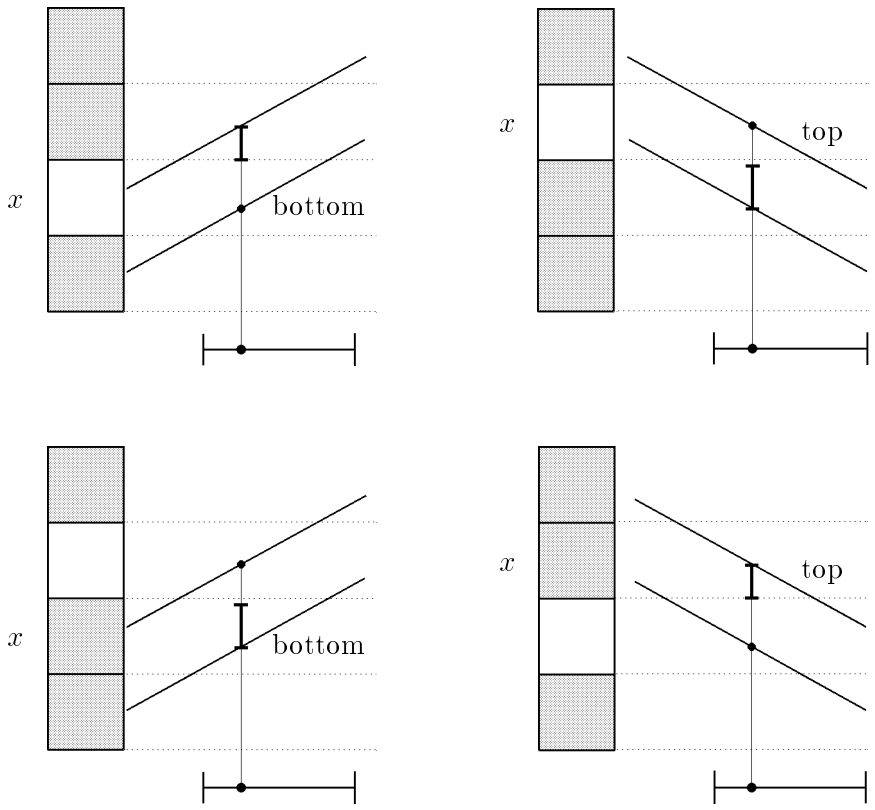


Abbildung 5.11: floor– und ceiling–constraints für Zielpixel x (begin–vlines)

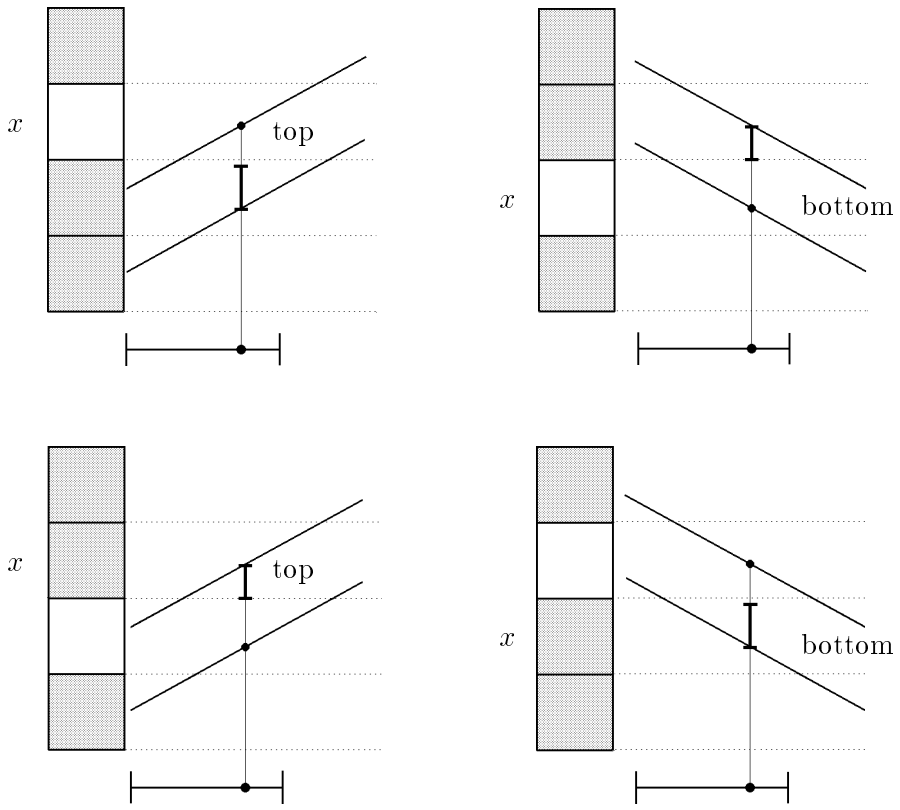


Abbildung 5.12: floor– und ceiling–constraints für Zielpixel x (end–vlines)

Symbol •) versehen. Die vier resultierenden constraints nennt man *Grenzconstraints*.

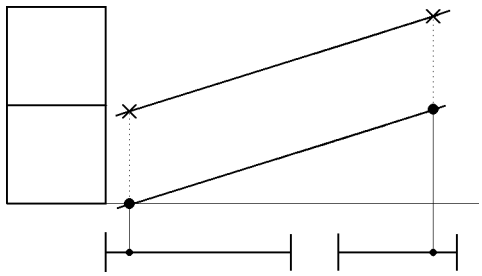


Abbildung 5.13: Die vier Grenzconstraints

In diesem Abschnitt wurden die vlines gezeigt, die, falls vorhanden, für ein Zielpixel relevant sind.

Das Vorhandensein oder Nichtvorhandensein der right–vlines und left–vlines für ein Zielpixel bedingt dessen weitere Behandlung.

5.1.6 Einfärben der Zielpixel

Mit dem Einfärben der Zielpixel ist eine Kategorisierung gemeint, keine tatsächliche Färbung. Diese Einteilung läßt keinen Schluß auf die tatsächliche Farbe des Pixels in G zu.

Kriterium ist die Konstellation der vlines im Zielpixelbereich. Der Zielpixelbereich eines Pixels x ist als grauer Bereich in Abb. 5.14 dargestellt.

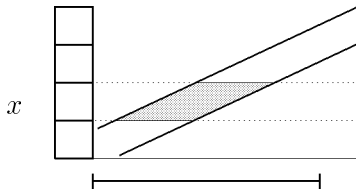


Abbildung 5.14: Der Zielpixelbereich eines Zielpixels x

Für jedes Zielpixel gibt es fünf Möglichkeiten:

1. WHITE
   Die relevanten vlines sind so angelegt, daß diese Pixel in G von keinem transformierten Mittelpunkt eines Pixels aus P getroffen werden können, egal wie die Lösungsgerade innerhalb von upper und lower gelegt wird (z.B. keine left–vline und keine right–vline im Zielpixelbereich). Die Pixel bleiben in jedem Fall weiß.

2. LIGHTGREY
   Es existiert eine right–vline im Zielpixelbereich. Abhängig von der Lösungsgerade kann das Pixel schwarz oder weiß werden.

3. GREY
   Es existiert eine left–vline im Zielpixelbereich. Abhängig von der Lösungsgerade kann das Pixel schwarz oder weiß werden.

4. DARKGREY
   Es existieren eine left–vline und eine right–vline im Zielpixelbereich. Einige dieser Pixel sind unabhängig von der Lösungsgerade immer schwarz, andere nicht. Das muß für jedes Pixel dieser Kategorie geprüft werden; die immer schwarzen Pixel werden dann der Kategorie BLACK zugeordnet. Die Prüfung ist unten beschrieben.

5. BLACK
   Diese Pixel werden auf jeden Fall von einem transformierten Mittelpunkt eines Pixels aus P getroffen, egal wie die Lösungsgerade innerhalb von upper und lower gelegt wird. Sie werden also sicher schwarzgefärbt. Es gibt zwei verschiedene Kriterien für Pixel dieser Klasse:

   - Es existieren eine left–vline und eine right–vline im Zielpixelbereich. Die constraints dieser vlines sind so angeordnet, daß diese Pixel in jedem Fall schwarz gefärbt werden, egal wie die Lösungsgerade innerhalb von upper und lower verläuft. Diese Kategorie erhält ihre Pixel durch eine Prüfung der Pixel aus der Kategorie DARKGREY.
   - Ein Dash wird unabhängig von der Lösungsgerade sicher auf diese Pixel abgebildet. Es gibt eine einfache Möglichkeit, diese Pixel für einen Dash zu bestimmen, die die Werte top und bottom der zugehörigen begin–vlines und end–vlines verwendet. Sie ist weiter unten beschrieben.

Die Prüfung der DARKGREY–Pixel

Wie bereits im Abschnitt 5.1.6 erwähnt, werden die Pixel der Klasse DARKGREY einer Sonderbehandlung unterzogen. Es wird festgestellt, ob es eine Lösungsgerade gibt, sodaß die entsprechende Transformation ein solches Pixel weiß läßt. Die Prüfung erfolgt über die constraints, die dem Pixel zugeordnet sind.

Abb. 5.15 zeigt ein DARKGREY–Pixel x im ungespiegelten Fall. Das Pixel ist im Zielmuster G weiß gefärbt; dementsprechend sind die constraints gesetzt. Sie erscheinen in der Abbildung fett gezeichnet.

Es wird nun die Gerade d_{min} mit der geringsten Steigung k_{dmin} ermittelt, die als Lösungsgerade das Pixel x weiß lassen würde. Sie ist durch zwei Punkte definiert, und zwar durch den floor–Punkt der left–vline und durch den ceiling–Punkt der right–vline.

Dann berechnet man d_{max}, die Gerade mit der größten Steigung k_{dmax}, die als Lösungsgerade x weiß ließe. Sie ist genauso durch zwei Punkte definiert, und zwar durch den bottom–Punkt

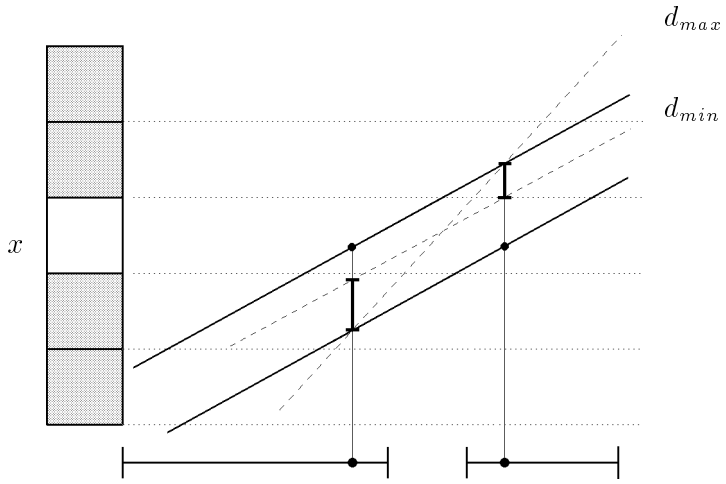


Abbildung 5.15: Die Geraden d_{min} und d_{max}

Gerade	definiert durch:
d_{min}	ceiling–Punkt der left–vline
	floor–Punkt der right–vline
d_{max}	top–Punkt der left–vline
	bottom–Punkt der right–vline

Tabelle 5.3: Definition der Geraden d_{min} und d_{max} im gespiegelten Fall

der left–vline und den top–Punkt der right–vline. Die Berechnung der beiden Geraden im gespiegelten Fall erfolgt über die Punkte in der Tabelle 5.3.

Zwei weitere Geraden werden ermittelt: f_{min} ist die Gerade mit der geringsten Steigung k_{fmin} innerhalb von upper und lower, f_{max} die Gerade mit der größten Steigung k_{fmax} innerhalb von upper und lower. Die Steigungen dieser vier Geraden werden nun miteinander verglichen.

Satz 5.9 *Ist $k_{fmax} < k_{dmin}$ oder $k_{dmax} < k_{fmin}$, dann fällt das betrachtete Pixel in die Kategorie BLACK.*

Beweis: Sind die Steigungen der Geraden d_{max}, d_{min}, f_{min} und f_{max} wie im Satz beschrieben, dann gibt es innerhalb von upper und lower keine Lösungsgerade, die x weiß lassen würde; das Pixel fällt somit in die Kategorie BLACK. Abb. 5.16 illustriert diesen Schluß. Die hier dargestellten Werte für k_{fmax} und k_{fmin} schließen einander natürlich aus; es kann nur jeweils einer dieser Werte auf der dargestellten Position existieren. □

Ist die obige Bedingung nicht erfüllt, verbleibt x in DARKGREY.

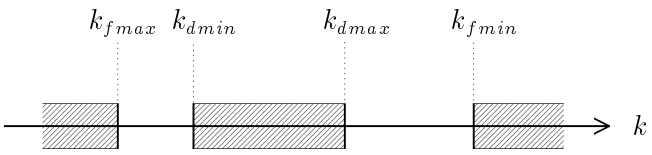


Abbildung 5.16: Haben die ermittelten Geraden solche Steigungen k, ist das betreffende Pixel sicher schwarz

Sichere Pixel der Klasse BLACK

Da der Bereich, in den die Lösungsgerade fallen kann, schon durch die Beschränkungsgeraden upper und lower eingegrenzt ist, ist es möglich, für jeden Dash über einer gewissen Minimallänge Pixel in G zu bestimmen, auf die sicher ein Teil des Dashs abgebildet wird. Diese Pixel werden unabhängig von der Lösungsgerade sicher schwarz sein. Die erwähnte Minimallänge wird später genau definiert; ist der Dash nämlich zu kurz, dann existieren für diesen keine sicher schwarzen Pixel.

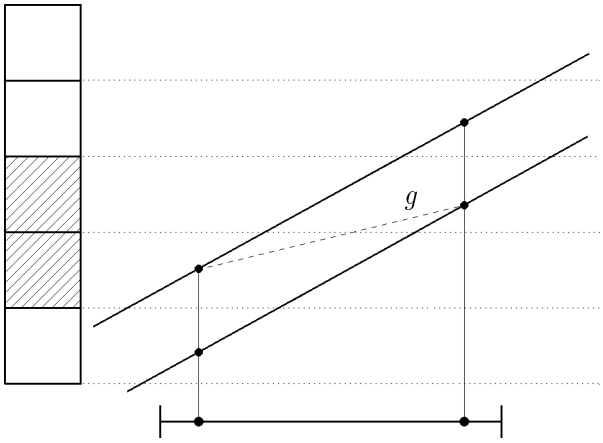


Abbildung 5.17: Die schraffierten Pixel werden unabhängig von w' sicher schwarz

Die Abb. 5.17 zeigt einen Dash samt vlines und Bereichen der Geraden upper und lower im ungespiegelten Fall. Er ist Teil eines Musters P. Die strichlierte Linie in der Abbildung zeigt ein Geradenstück g, das durch den top–Punkt der begin–vline und durch den bottom–Punkt der end–vline definiert ist. Es hat sicher eine geringere Steigung als die flachste mögliche Lösungsgerade. Das heißt, daß jede mögliche Lösungsgerade den Dash auf mindestens diejenigen Pixel in P abbildet, deren Zielpixelbereich von g geschnitten wird. Sie sind in der Abbildung schraffiert gezeichnet statt grau wie sonst, weil hier die von der Gerade g implizierte Farbe und nicht die Farbe der Zielpixel gezeigt wird. Diese Pixel haben die Indizes top der begin–vline bis

bottom der end–vline. Sie fallen in die Kategorie BLACK.

Im gespiegelten Fall verläuft die Berechnung ähnlich. Hier haben die Pixel die Indizes bottom der begin–vline bis top der end–vline.

Damit solche Pixel für einen Dash existieren, muß dieser mindestens so lang sein, daß im ungespiegelten Fall der Index top der begin–vline $\leq$ dem Index bottom der end–vline ist und im gespiegelten Fall der Index bottom der begin–vline $\geq$ dem Index top der end–vline.

Vergleich der BLACK– und der WHITE–Pixel mit G

Im Gegensatz zu den Pixeln in den Klassen LIGHTGREY, GREY und DARKGREY sind die Pixel in BLACK bzw. WHITE unabhängig von der Lösungsgerade schwarz bzw. weiß. Deshalb werden sie sofort mit den Pixeln des Zielmusters G verglichen. Ist ein Pixel der Klasse BLACK in G nicht schwarz bzw. ein Pixel der Klasse WHITE in G nicht weiß, so ist es unmöglich, eine Transformation w' zu finden, die P auf G abbildet. Der Prozeß der Lösungsfindung wird abgebrochen. Ist der Vergleich erfolgreich, wird mit der Bestimmung der relevanten constraints fortgesetzt.

5.1.7 Relevante constraints

Für die Zielpixel der Klassen LIGHTGREY, GREY und DARKGREY müssen nun die relevanten constraints ermittelt werden, die die endgültige Beschränkung der Lösungsgerade ergeben. Eine Lösungsgerade, die alle relevanten constraints schneidet, entspricht einem w', das das gewünschte Pixelmuster G erzeugt.

Aus den Ergebnissen des Abschnittes 5.1.5 und der Tatsache, daß ein LIGHTGREY–Pixel nur eine right–vline im Zielpixelbereich hat, ein GREY–Pixel nur eine left–vline, ein DARKGREY–Pixel dagegen beides, wurde die Tabelle 5.4 erstellt: Die Eintragung der Zeile „LIGHTGREY“ unter der Rubrik „Schwarzes Zielpixel – ungespiegelt“ bedeutet z.B. folgendes: Bei einem schwarzen Zielpixel aus G, das sich in der Klasse LIGHTGREY befindet, also nur eine right–vline im Zielpixelbereich hat, ist der relevante constraint ein floor–constraint, d.h. der Bereich, den die Lösungsgerade schneiden muß, reicht auf der right–vline von y_{bottom} bis y_{floor}.

Die Eintragung OK heißt, daß dieses Pixel bei der Berechnung der Lösungsgerade nicht mehr beachtet werden muß, da seine Farbe bereits fix ist und mit der gewünschten Farbe in G übereinstimmt, egal wie die Lösungsgerade innerhalb upper und lower ausfällt.

Der Eintrag FAIL bedeutet, daß die Farbe dieses Pixels sicher nicht mit seiner in G gewünschten Farbe übereinstimmt, und zwar unabhängig von der Lösungsgerade. Der Prozeß kann in diesem Fall abgebrochen werden, da es keine Lösung gibt (siehe auch den Absatz „Vergleich der BLACK– und der WHITE–Pixel mit G“ im Abschnitt 5.1.6). Auf die Eintragungen in der Zeile DARKGREY wird später noch genauer eingegangen.

Es sei noch erwähnt, daß neben den constraints in der Tabelle natürlich auch die Grenzconstraints relevant sind.

	Schwarzes Zielpixel		Weißes Zielpixel	
	ungespiegelt	gespiegelt	ungespiegelt	gespiegelt
LIGHTGREY	right–vline floor–c.	right–vline ceiling–c.	right–vline ceiling–c.	right–vline floor–c.
GREY	left–vline ceiling–c.	left–vline floor–c.	left–vline floor–c.	left–vline ceiling–c.
DARKGREY	right–vline floor–c. oder left–vline ceiling–c.	right–vline ceiling–c. oder left–vline floor–c.	right–vline ceiling–c. und left–vline floor–c.	right–vline floor–c. und left–vline ceiling–c.
WHITE	FAIL		OK	
BLACK	OK		FAIL	

Tabelle 5.4: Relevante constraints

5.1.8 Beschreibung der Lösungsgerade

Aus den somit für alle Zielpixel in G festgelegten constraints kann man eine Beschreibung der Lösungsgerade ableiten. Es handelt sich um eine Gerade, die (in y–Richtung) unter oder auf den floor–Punkten aller relevanten floor–constraints und über oder auf den ceiling–Punkten aller relevanten ceiling–constraints verläuft, d.h. die alle relevanten floor– und ceiling–constraints schneidet. Man sagt von einer solchen Gerade, daß sie alle relevanten constraints erfüllt.

Diese Gerade verläuft wegen der Grenzconstraints sicher innerhalb der Beschränkungsgeraden upper und lower. Abb. 5.18 veranschaulicht diese Tatsache. Wie üblich wird das Symbol • für

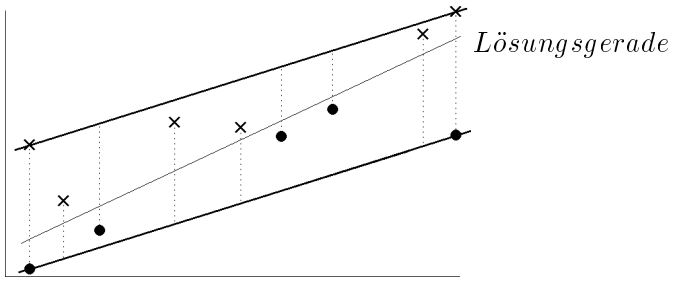


Abbildung 5.18: Eine gültige Lösungsgerade

ceiling–Punkte und das Symbol × für floor–Punkte verwendet. Eine solche Gerade bildet das Pixelmuster P in der im Abschnitt 5.1.1 beschriebenen Weise korrekt auf das Zielpixelmuster G ab. Durch diese Gerade wird die gesuchte Transformation w definiert.

5.2 Theoretische Auffindung der Lösungsgerade

Es wird gezeigt, wie man eine Lösungsgerade findet, die die Bedingungen des Abschnitts 5.1.8 erfüllt.

5.2.1 Die Beschränkungsgeraden b_{min} und b_{max}

Es werden zwei Geraden bestimmt: b_{min} ist die steilste Gerade, die einen floor–Punkt mit einem ceiling–Punkt verbindet, wobei der floor–Punkt links vom ceiling–Punkt liegen muß.

Die zweite Gerade b_{max} ist die flachste Gerade, die einen ceiling–Punkt mit einem floor–Punkt verbindet, wobei der ceiling–Punkt links vom floor–Punkt liegen muß.

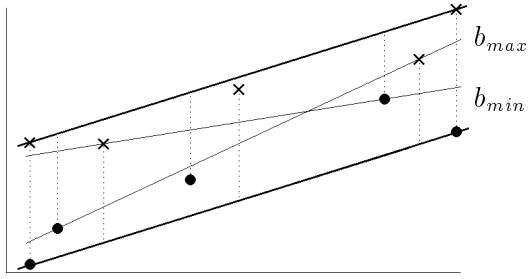


Abbildung 5.19: Die Geraden b_{min} und b_{max}

Abb. 5.19 zeigt je ein Beispiel für b_{min} und b_{max}. Wir betrachten nun die Steigungen k der Geraden. Ist $k_{bmin} \leq k_{bmax}$, sind beide Geraden Lösungsgeraden. Ein Beweis für diese Behauptung ist im Abschnitt 5.2.2 angeführt. In der Abb. 5.19 sind beide Geraden Lösungsgeraden; man sieht, daß hier alle constraints erfüllt sind.

Ist jedoch $k_{bmin} > k_{bmax}$, so gibt es keine Lösungsgerade, die alle constraints erfüllt, und somit keine Transformation w', die P diskret auf G abbildet.

5.2.2 Die Trennung zweier Punktmengen durch eine Gerade

In diesem Abschnitt wird dargelegt, wie man entscheiden kann, ob zwei Punktmengen $\in \mathbb{R}^2$ durch eine Gerade getrennt werden können. Falls die Möglichkeit einer Separation besteht, kann über das Kriterium eine Menge von Separationsgeraden ermittelt werden.

Problemdefinition

Vereinbarungen:

- $\mathcal{K}_A$:= Konvexe Hülle der Menge $A \subset \mathbb{R}^2$
- A° := das Innere von A
- $\tilde{A}$:= der Abschluß von A
- δA := der Rand von $A := \tilde{A} \setminus A^\circ$

$\mathcal{K}_A$ ist eine kompakte (d.h. abgeschlossene und beschränkte) Menge, wenn A beschränkt ist. Der Abschluß, das Innere und der Rand von A werden in der Abb. 5.20 illustriert.

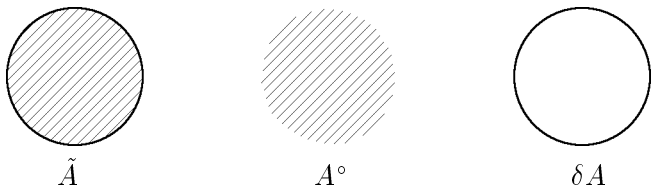


Abbildung 5.20: Der Abschluß, das Innere und der Rand von A

Es sind zwei Punktmengen $\subset \mathbb{R}^2$ gegeben:

$F \ldots$ die Menge der floor–Punkte
$C \ldots$ die Menge der ceiling–Punkte

Das Symbol $\times$ wird wie in früheren Kapiteln für $c \in C$ verwendet, das Symbol $\bullet$ für $f \in F$. Diese beiden Punktmengen werden folgendermaßen eingeschränkt:

Es gibt $f_l, f_r \in F$ und $c_l, c_r \in C$, sodaß f_l und c_l bzw. f_r und c_r jeweils dieselbe x–Koordinate haben; f_l und f_r liegen in y–Richtung höher als c_l und c_r. Weiters liegen f_l und c_l links von f_r und c_r. Die konvexe Hülle $\mathcal{K}_{fl,fr,cl,cr}$ dieser vier Punkte enthält alle Punkte aus F und alle Punkte aus C. F und C sind unter Berücksichtigung dieser Einschränkungen in Abb. 5.21 dargestellt.

Die Frage ist nun, ob es eine Gerade gibt, die F und C separiert.

Zusammenhang der Frage mit konvexen Hüllen

Man kann die Frage nach einer Separationsgerade für F und C mit Hilfe von konvexen Hüllen von F und C beantworten.

Satz 5.10 *Es gibt eine Separationsgerade genau dann, wenn* $\mathcal{K}_F{}^\circ \cap \mathcal{K}_C{}^\circ = \emptyset$.

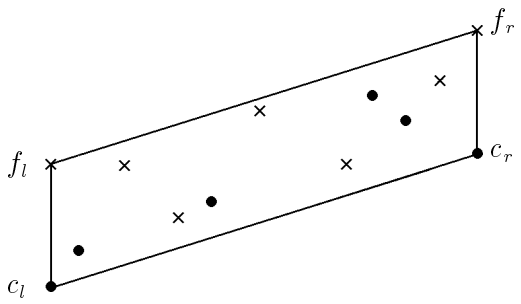


Abbildung 5.21: Einschränkungen für die Mengen F und C

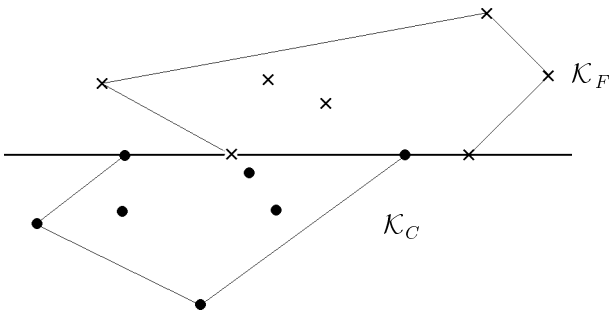


Abbildung 5.22: Die konvexen Hüllen $\mathcal{K}_F$ und $\mathcal{K}_C$

Beweis:

1. $\exists$ Separationsgerade $\Rightarrow {\mathcal{K}_F}^\circ \cap {\mathcal{K}_C}^\circ = \emptyset$
   Gibt es eine Separationsgerade, dann können sich $\mathcal{K}_F$ und $\mathcal{K}_C$ maximal auf dieser Gerade schneiden (Abb. 5.22). Dann schneiden sich nur $\delta\mathcal{K}_F$ und $\delta\mathcal{K}_C$, nicht aber ${\mathcal{K}_F}^\circ$ und ${\mathcal{K}_C}^\circ$.

2. ${\mathcal{K}_F}^\circ \cap {\mathcal{K}_C}^\circ = \emptyset \Rightarrow \exists$ Separationsgerade

   2.1 $\mathcal{K}_F \cap \mathcal{K}_C = \emptyset$
   Eine Separationsgerade existiert trivialerweise.

   2.2 $\tilde{\mathcal{K}_F} \cap \tilde{\mathcal{K}_C} \neq \emptyset$
   Wenn $\tilde{\mathcal{K}_F} \cap \tilde{\mathcal{K}_C} \neq \emptyset$, aber ${\mathcal{K}_F}^\circ \cap {\mathcal{K}_C}^\circ = \emptyset$, dann können sich nur $\delta\mathcal{K}_F$ und $\delta\mathcal{K}_C$ schneiden, nicht aber ${\mathcal{K}_F}^\circ$ und ${\mathcal{K}_C}^\circ$. Die möglichen Fälle sind in der Abb. 5.23 dargestellt. Man sieht, daß immer eine Separationsgerade gezogen werden kann.

□

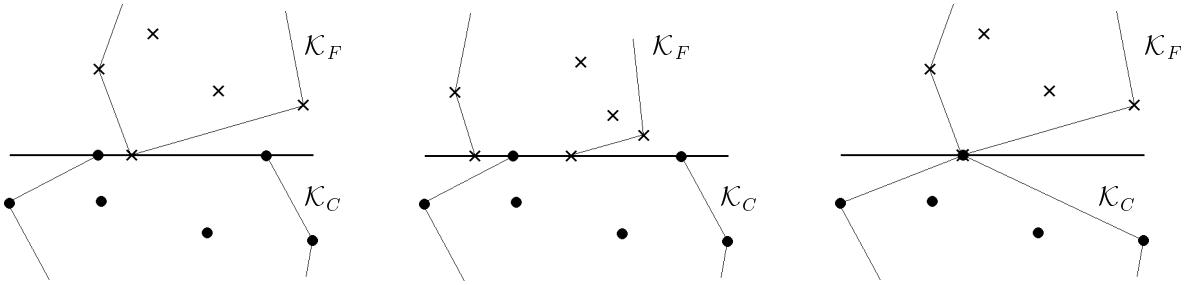


Abbildung 5.23: Die möglichen Lagen der konvexen Hüllen $\mathcal{K}_F$ und $\mathcal{K}_C$

Das Kriterium

Die Geraden b_{min} und b_{max} wurden im Abschnitt 5.2.1 erklärt. Hier eine formale Definition:

Definition 5.11 $\overline{FC} := \{\ \overline{fc}\ |f \in F, c \in C, x_f \leq x_c\ \}$
$\overline{CF} := \{\ \overline{cf}\ |c \in C, f \in F, x_c \leq x_f\ \}$

$b_{min} \in \overline{FC} : \forall g \in \overline{FC} : k_{bmin} \geq k_g$
$b_{max} \in \overline{CF} : \forall g \in \overline{CF} : k_{bmax} \leq k_g$

Hier ist $\overline{FC}$ die Menge der Geraden, die durch die Punkte $f \in F, c \in C$ definiert sind, wobei f links von c ist. $\overline{CF}$ ist die Menge der Geraden, die durch die Punkte $c \in C, f \in F$ definiert sind, wobei c links von f ist. Die Punkte, durch die b_{min} definiert ist, werden f_{min} und c_{min} genannt. Die Punkte, durch die b_{max} definiert ist, werden c_{max} und f_{max} genannt.

Eigenschaften der Geraden b_{min} und b_{max}

Aus den obigen Definitionen ergeben sich Bereiche in $\mathbb{R}^2$, in denen sich keine floor–Punkte bzw. ceiling–Punkte aufhalten dürfen.

Hilfssatz 5.12 *Rechts von und auf c_{max} liegt kein $f \in F$ unter b_{max}.*
Links von und auf f_{max} liegt kein $c \in C$ über b_{max}.
Rechts von und auf f_{min} liegt kein $c \in C$ über b_{min}.
Links von und auf c_{min} liegt kein $f \in F$ unter b_{min}.

Diese Bereiche sind in Abb. 5.24 dargestellt.

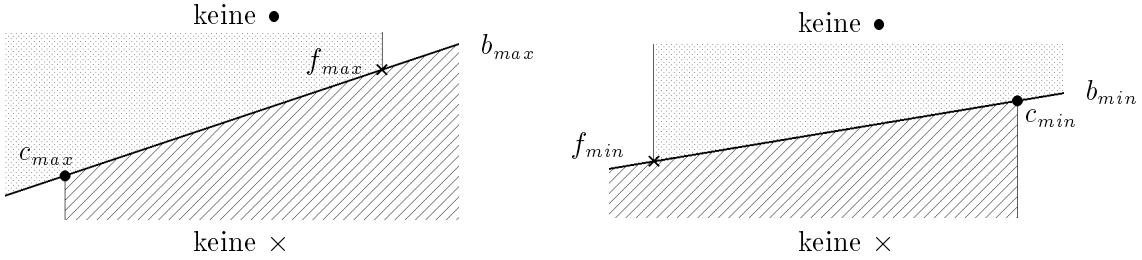


Abbildung 5.24: Verbotene Bereiche für × und •

Beweis: Wäre rechts von oder auf c_{max} ein $f \in F$ unter b_{max}, dann wäre $\overline{c_{max}f}$ flacher als b_{max}, was einen Widerspruch zur Definition von b_{max} bedeutet. Wäre links von oder auf f_{max} ein $c \in C$ über b_{max}, dann wäre $\overline{cf_{max}}$ flacher als b_{max}, was einen Widerspruch zur Definition von b_{max} bedeutet. Wäre rechts von oder auf f_{min} ein $c \in C$ über b_{min}, dann wäre $\overline{f_{min}c}$ steiler als b_{min}, was einen Widerspruch zur Definition von b_{min} bedeutet. Wäre links von oder auf c_{min} ein $f \in F$ unter b_{min}, dann wäre $\overline{fc_{min}}$ steiler als b_{min}, was einen Widerspruch zur Definition von b_{min} bedeutet. □

Hilfssatz 5.13 *Wenn eine Separationsgerade existiert, dann gilt, daß b_{min} und b_{max} Separationsgeraden sind und daß $k_{bmin} \leq k_{bmax}$.*

Beweis:

1. $\exists$ Separationsgerade $\Rightarrow b_{min}$ und b_{max} sind Separationsgeraden.
   Damit eine Gerade b eine Separationsgerade ist, darf über b kein $c \in C$ und unter b kein $f \in F$ sein. Aus Hilfssatz 5.12 ist bekannt, daß diese Bedingung bei b_{min} und b_{max} für einige Teilgebiete von $\mathbb{R}^2$ erfüllt ist.

   Betrachten wir zunächst b_{max}. Wie auf Abb. 5.24 zu erkennen ist, müssen nur noch das Gebiet rechts von f_{max} über b_{max} sowie das Gebiet links von c_{max} unter b_{max} überprüft werden.

   Angenommen, es gibt ein $c \in C$ rechts von f_{max} über b_{max}. Laut Satz 5.10 gibt es eine Separationsgerade für C und F genau dann, wenn sich die konvexen Hüllen von C und F nicht schneiden.

   Bildet man nun wie in Abb. 5.25 die konvexe Hülle für C und F, indem man die in der Problemstellung definierten Randpunkte hinzunimmt, dann sieht man, daß sich die konvexen Hüllen schneiden. Das ist ein Widerspruch zu den angenommenen Voraussetzungen. Es kann also kein solches $c \in C$ geben.

   Der Beweis für das Gebiet links von c_{max} unter b_{max} verläuft sinngemäß. Die konvexen Hüllen für diesen Fall sind in Abb. 5.26 dargestellt. Es kann also auch kein $f \in F$ links von c_{max} unter b_{max} geben.

   Entsprechend kann man den Beweis auch für b_{min} führen. Damit ist bewiesen, daß b_{min} und b_{max} Separationsgeraden sind.

2. $\exists$ Separationsgerade $\Rightarrow k_{bmin} \leq k_{bmax}$.
   Im ersten Teil dieses Beweises wurde gezeigt, daß es über einer Separationsgerade keine $c \in C$ und unter einer Separationsgerade keine $f \in F$ geben kann. Betrachten wir eine fixe Gerade b_{max}. f_{min} kann nur auf oder über b_{max} liegen, c_{min} nur auf oder unter b_{max}. Außerdem muß nach der Definition von b_{min} $x_{fmin} \leq x_{cmin}$ sein. Daraus folgt, daß $k_{bmin} \leq k_{bmax}$.

□

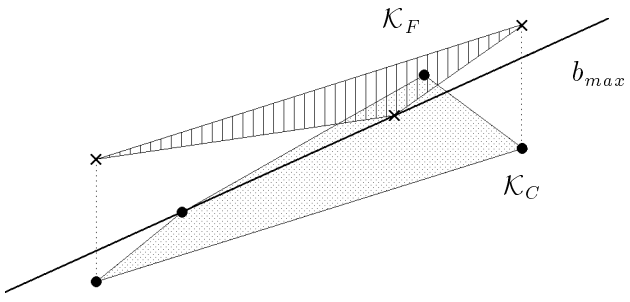


Abbildung 5.25: Die konvexen Hüllen $\mathcal{K}_F$ und $\mathcal{K}_C$ schneiden sich

Im Hilfssatz 5.15 wird bewiesen werden, daß b_{min} und b_{max} Separationsgeraden sind, wenn $k_{bmin} \leq k_{bmax}$ gilt. Dazu ist es notwendig, weitere Bereiche in $\mathbb{R}^2$ zu definieren, in denen keine floor–Punkte bzw. ceiling–Punkte liegen dürfen. Zur Vereinfachung des Beweises für den

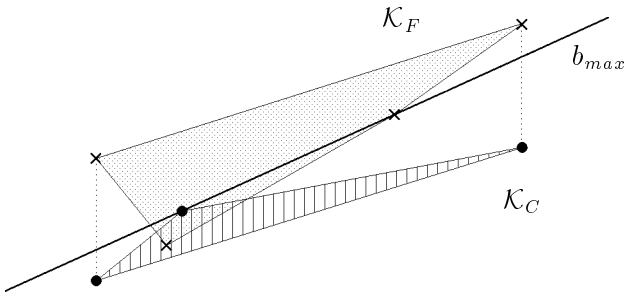


Abbildung 5.26: Die konvexen Hüllen $\mathcal{K}_F$ und $\mathcal{K}_C$ schneiden sich

Hilfssatz 5.15 wird ein Teil dieser für floor–Punkte bzw. ceiling–Punkte verbotenen Bereiche im Hilfssatz 5.14 ermittelt. Der Hilfssatz 5.14 wird dann im Beweis des Hilfssatzes 5.15 verwendet.

Hilfssatz 5.14 *Es gilt* $x_{fmin} \leq x_{fmax}$ *sowie* $x_{cmin} \geq x_{cmax}$.

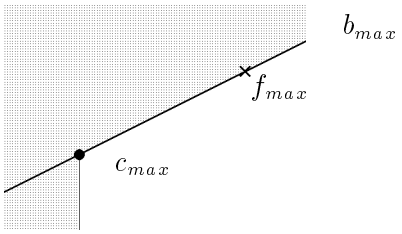


Abbildung 5.27: f_{min} muß im grauen Bereich liegen

Beweis: Betrachten wir wiederum ein fixes b_{max}. Abb. 5.27 zeigt die Bereiche, in denen f_{min} aufgrund von Hilfssatz 5.12 liegen muß. Es stellt sich die Frage, in welchen dieser Bereiche f_{min} *nicht* liegen darf, damit der Hilfssatz garantiert ist. Der fragliche Bereich ist derjenige rechts von f_{max} über b_{max}.

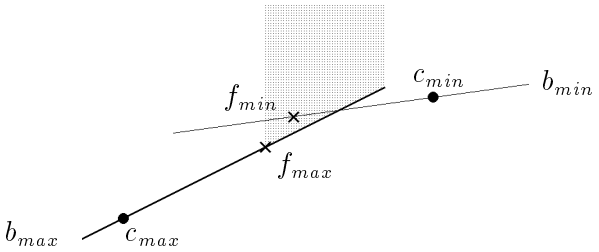


Abbildung 5.28: Annahme: f_{min} liegt rechts von f_{max} über b_{max}

Angenommen, f_{min} befindet sich in diesem Bereich. Dieser Fall ist in Abb. 5.28 gezeigt. Durch die Voraussetzung $k_{bmin} \leq k_{bmax}$ ergibt sich, daß f_{max} auf jeden Fall unter b_{min} liegt. c_{min}

ist laut Definition der Gerade b_{min} rechts von f_{min} und damit auch rechts von f_{max}. Laut Hilfssatz 5.12 gilt jedoch, daß links von c_{min} kein $f \in F$ unter b_{min} liegen darf. Das ist ein Widerspruch zur Lage von f_{max}. f_{min} kann also nicht im Bereich rechts von f_{max} über b_{max} liegen.

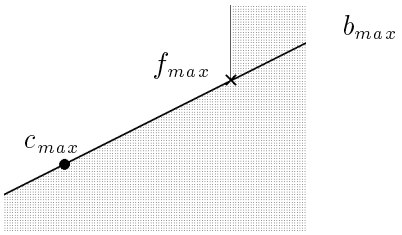


Abbildung 5.29: c_{min} muß im grauen Bereich liegen

Nun wird die Lage von c_{min} überprüft. Abb. 5.29 zeigt die Bereiche, in denen c_{min} aufgrund von Hilfssatz 5.12 liegen muß. Der Bereich, in denen c_{min} *nicht* liegen darf, ist derjenige links von c_{max} unter b_{max}. Ähnlich wie für f_{min} zeigt man, daß c_{min} nicht in diesem Bereich liegen kann.

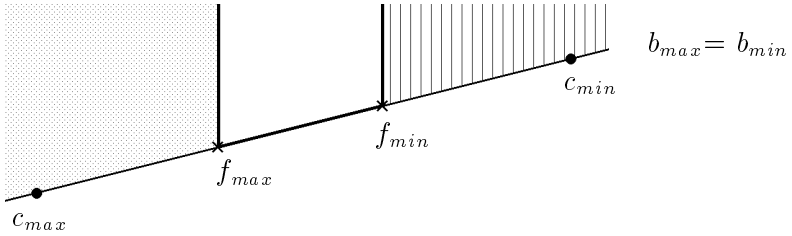


Abbildung 5.30: Der Sonderfall $b_{min} = b_{max}$

Der Fall $b_{min} = b_{max}$ ist ein Sonderfall. Hier kann die obige Argumentation nicht angewendet werden; es ist eine Situation wie in Abb. 5.30 möglich. In diesem Fall werden die Werte von f_{min} und f_{max} vertauscht, was erlaubt ist. □

Hilfssatz 5.15 *Ist $k_{bmin} \leq k_{bmax}$, dann sind b_{min} und b_{max} Separationsgeraden.*

Beweis: Damit eine Gerade b eine Separationsgerade ist, darf über b kein $c \in C$ und unter b kein $f \in F$ sein. Aus Hilfssatz 5.12 ist bekannt, daß diese Bedingung bei b_{min} und b_{max} für einige Teilgebiete von $\mathbb{R}^2$ erfüllt ist. Fügen wir b_{min} und b_{max} so zusammen, daß $k_{bmin} \leq k_{bmax}$, sehen wir, daß b_{min} und b_{max} den Kriterien für eine Separationsgerade entsprechen (Abb. 5.31).

Das ist jedoch nur der Fall, wenn sich der Schnittpunkt von b_{min} und b_{max} im Bereich der Geradenstücke $\overline{f_{min}c_{min}}$ bzw. $\overline{c_{max}f_{max}}$ befindet. Für einen anderen Schnittpunkt müssen die Kriterien für eine Separationsgerade nicht erfüllt sein (Abb. 5.32). Es muß also ausgeschlossen werden, daß ein solcher unzulässiger Schnittpunkt auftritt.

Nach Hilfssatz 5.14 gilt $x_{fmin} \leq x_{fmax}$ sowie $x_{cmin} \geq x_{cmax}$. Betrachten wir nun nochmals die Abb. 5.27. Eine weitere Annahme für f_{min}, die einen unzulässigen Schnittpunkt ergibt und die

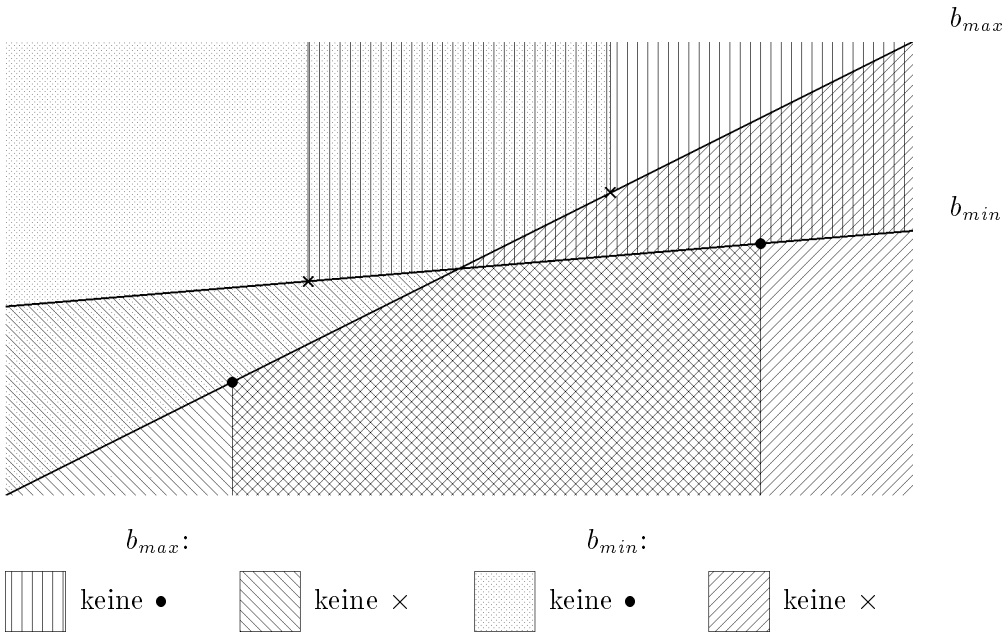


Abbildung 5.31: Hier sind b_{min} und b_{max} Separationsgeraden

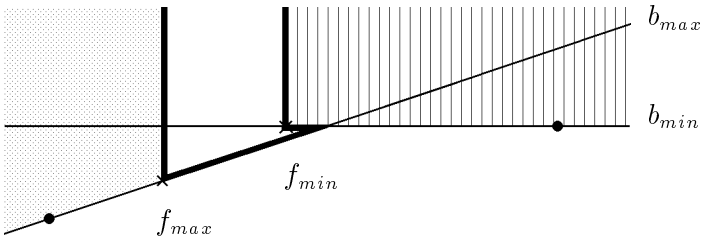


Abbildung 5.32: Für den fett umrandeten Bereich kann nicht garantiert werden, daß dort keine • liegen

noch nicht ausgeschlossen wurde, ist die folgende: f_{min} liegt links von c_{max} und unter b_{max}, sodaß wegen $k_{bmin} \leq k_{bmax}$ der Schnittpunkt unzulässig ist.

Dann würde rechts von f_{min} ein $c \in C$ über b_{min} liegen; das ist ein Widerspruch zu Hilfssatz 5.12.

Dieses Ergebnis zusammen mit Hilfssatz 5.14 und Hilfssatz 5.12 ergibt einen Bereich, in dem f_{min} liegen muß; er ist in Abb. 5.33 dargestellt.

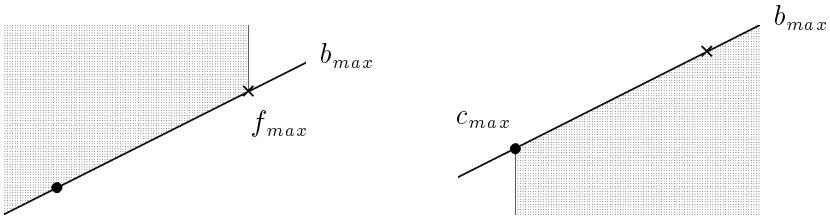


Abbildung 5.33: Mögliche Bereiche für f_{min} (links) und c_{min} (rechts)

Auf ähnliche Weise wird ein Bereich ermittelt, in dem c_{min} liegen muß; er ist ebenfalls in Abb. 5.33 dargestellt.

Sämtliche Fälle für einen ungültigen Schnittpunkt, die unter den Beschränkungen von Abb. 5.33 noch denkbar sind, werden durch Hilfssatz 5.12 ausgeschlossen. Daraus ergibt sich, daß sich der Schnittpunkt von b_{min} und b_{max} im Bereich der Geradenstücke $\overline{f_{min}c_{min}}$ bzw. $\overline{c_{max}f_{max}}$ befinden muß.

Damit ist bewiesen, daß b_{min} und b_{max} Separationsgeraden sind, wenn $k_{bmin} \leq k_{bmax}$. Denn über den beiden Geraden können sich keine $c \in C$ befinden und unter ihnen keine $f \in F$, wie die Abb. 5.31 zeigt. □

Satz 5.16 $k_{bmin} \leq k_{bmax} \Leftrightarrow \exists$ *Separationsgerade*

Beweis:

1. $k_{bmin} \leq k_{bmax} \Rightarrow \exists$ Separationsgerade
   Mit Hilfssatz 5.15 sind b_{min} und b_{max} Separationsgeraden, also gibt es eine Separationsgerade.

2. $\exists$ Separationsgerade $\Rightarrow k_{bmin} \leq k_{bmax}$
   Mit Hilfssatz 5.13 wahr.

□

Satz 5.17 $\exists$ *Separationsgerade* $\Leftrightarrow$ b_{min} *und* b_{max} *sind Separationsgeraden*

Beweis:

1. $\exists$ Separationsgerade $\Rightarrow$ b_{min} und b_{max} sind Separationsgeraden
   Mit Hilfssatz 5.13 wahr.

2. b_{min} und b_{max} sind Separationsgeraden $\Rightarrow \exists$ Separationsgerade
   trivial

□

5.3 Praktische Auffindung der Lösungsgerade

Wir gehen davon aus, daß die relevanten constraints durch die Tabelle 5.4 im Abschnitt 5.1.7 gefunden sind. Aus diesen constraints sollen die Geraden b_{min} und b_{max} und damit eine Lösungsgerade ermittelt werden.

5.3.1 AND– und OR–constraints

Aus der Tabelle 5.4 im Abschnitt 5.1.7 ist ersichtlich, daß die Pixel der Klasse DARKGREY eine gewisse Sonderstellung einnehmen. Ist ein Pixel dieser Klasse in G schwarz, muß nur einer der beiden zugehörigen constraints erfüllt sein; es genügt ja, daß das Zielpixel unter w' von einem Punkt getroffen wird, damit es schwarz gezeichnet wird. Ist dagegen ein Pixel dieser Klasse in G weiß, müssen beide zugehörigen constraints erfüllt sein, denn das Pixel darf unter w' nie getroffen werden.

Daraus ergibt sich, daß zwei Arten von constraints existieren:

1. AND–constraints
   Jeder einzelne dieser constraints muß erfüllt sein. Die constraints von Pixeln der Kategorien LIGHTGREY und GREY sind AND–constraints. Die constraints von Pixeln der Klasse DARKGREY sind bei weißen Zielpixeln ebenfalls AND–constraints; es gibt jedoch hier zwei constraints pro Pixel. Sie können wie alle anderen AND–constraints behandelt werden.

2. OR–constraints
   Eine OR–constraint—Gruppe ist eine Gruppe von zwei constraints, von denen nur einer erfüllt sein muß. Ein constraint aus einer solchen Gruppe ist ein OR–constraint. Die constraints von Pixeln der Klasse DARKGREY sind bei schwarzen Zielpixeln OR–constraints.

Aus dieser Unterteilung in AND–constraints und OR–constraints ergibt sich die Darstellung eines gerichteten Graphen (Abb. 5.34), in der die relevanten constraints geordnet nach x–Werten als Knoten von links nach rechts aufgeführt sind. Die weißen Knoten in der Abb. 5.34 stellen die AND–constraints dar, die grauen Knoten die OR–constraints, wobei immer zwei übereinanderstehende Knoten eine OR–constraint–Gruppe ergeben. Wir werden diesen Graphen als AND/OR–constraint–Kette bezeichnen. Die Suche nach einer Lösungsgerade entspricht in dieser Darstellung einem Weg von einem Ende des Graphen zum anderen, in der Abb. 5.34 also einem Weg vom Knoten A zum Knoten J oder von J nach A. Jeder constraint, der passiert wird, muß erfüllt werden.

Da die Reihenfolge, in der die constraints geprüft werden, irrelevant ist, kann die Prüfung gut auf Parallelrechnern implementiert werden. Hier wird jedoch eine sequentielle Implementierung vorgestellt.

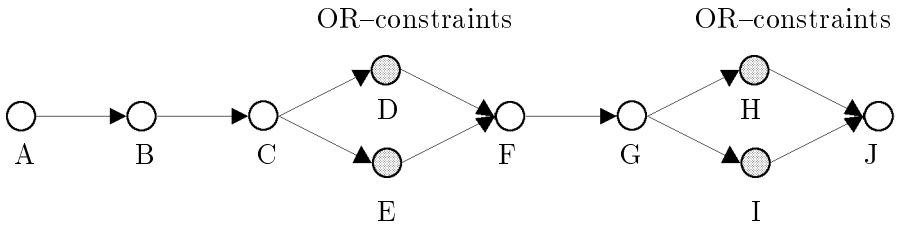


Abbildung 5.34: Eine AND/OR–constraint–Kette

Verringerung des Suchaufwandes

Sei m die Anzahl der OR–constraint–Gruppen in einer AND/OR–constraint–Kette. Dann gibt es 2^m verschiedene Wege durch den Graphen.

Nehmen wir an, daß in einer AND/OR–constraint–Kette zwei AND–constraints einander widersprechen, sodaß keine Lösungsgerade möglich ist. Würde man zur Lösungsfindung immer alle Wege prüfen, dann würde ein Mißerfolg erst nach der Prüfung des 2^m–ten Weges feststehen.

Deshalb wird eine andere Strategie verfolgt: Zunächst werden die OR–constraints vernachlässigt und nur die AND–constraints geprüft, d.h. so bei der Suche nach b_{min} und b_{max} berücksichtigt, daß sie erfüllt sind. Im Beispiel in der Abb. 5.34 sind das die constraints, die in der Abb. 5.35 gezeigt werden. Erst wenn dieser Schritt erfolgreich war, werden zusätzlich die OR–constraints geprüft.

Die Prüfung der OR–constraints erfolgt mittels Backtracking. Hier ein Beispiel anhand der Abb. 5.34: Es wird aus jeder OR–constraint—Gruppe ein OR–constraint gewählt und für diese OR–constraints zusammen mit allen AND–constraints eine Lösungsgerade gesucht. Im Beispiel der Abb. 5.34 wählt man z.B. die OR–constraints D und H und sucht für die constraints A, B, C, D, F, G, H und J eine Lösungsgerade. Ist keine Lösungsgerade möglich, wählt man statt dem OR–constraint H den OR–constraint I und sucht für die constraints A, B, C, D, F, G, I und J eine Lösungsgerade. Ist wiederum keine Lösungsgerade möglich, wählt man statt dem OR–constraint D den OR–constraint E und statt dem OR–constraint I wieder den OR–constraint H und sucht für die constraints A, B, C, E, F, G, H und J eine Lösungsgerade. Im Fall eines Mißerfolgs sucht man für die letzte mögliche Kombination A, B, C, E, F, G, I und J eine Lösungsgerade. Somit werden mittels Backtracking alle möglichen Wege im Graph geprüft.

5.3.2 Die Prüfung der AND–constraints

Zur Prüfung der AND–constraints wird die AND/OR–constraint–Kette verändert. Dazu werden die OR–constraints sowie deren einlaufende und wegführende Kanten eliminiert und die AND–constraints dahinter und davor mit einer Kante verbunden. Das Resultat ist eine AND–constraint–Kette. Die in der Abb. 5.35 dargestellte AND–constraint–Kette ergibt sich aus der AND/OR–constraint–Kette in Abb. 5.34. Der folgende Algorithmus **Check_AND_constraints**

Abbildung 5.35: Eine AND–constraint–Kette

durchläuft die AND–constraint–Kette von links nach rechts und versucht dabei, die Geraden b_{min} und b_{max} zu bestimmen, sodaß die bereits passierten Knoten (AND–constraints) erfüllt sind. b_{min} und b_{max} werden entsprechend initialisiert.

Check_AND_constraints (AND–constraint–Kette)
Folgendes wird für alle Knoten r von links nach rechts durchgeführt:

- Ist r ein floor–constraint und c dessen floor–Punkt, dann wird
  - Folgendes für alle Knoten s von ganz rechts bis exkl. r durchgeführt:
    - Ist s ein ceiling–constraint und d dessen ceiling–Punkt, dann wird die Steigung k_{bmin} mit der Steigung k der Gerade durch c und d verglichen. Ist $k > k_{bmin}$, dann b_{min}:= die Gerade durch c und d.
      Ist nun $k_{bmin} \leq k_{bmax}$, dann ist eine Lösungsgerade noch möglich; sonst wird der Algorithmus abgebrochen.
- Ist r ein ceiling–constraint und c dessen ceiling–Punkt, dann wird
  - Folgendes für alle Knoten s von ganz links bis exkl. r durchgeführt:
    - Ist s ein floor–constraint und d dessen floor–Punkt, dann wird die Steigung k_{bmax} mit der Steigung k der Gerade durch d und c verglichen. Ist $k < k_{bmax}$, dann b_{max}:= die Gerade durch d und c.
      Ist nun $k_{bmin} \leq k_{bmax}$, dann ist eine Lösungsgerade noch möglich; sonst wird der Algorithmus abgebrochen.

Wird der Algorithmus nicht frühzeitig abgebrochen, sondern terminiert ordnungsgemäß, so bedeutet das folgendes: b_{min} und b_{max} wurden gefunden, sodaß $k_{bmin} \leq k_{bmax}$, und b_{min} und b_{max} erfüllen alle AND–constraints der AND–constraint–Kette. Es gibt also (mindestens) eine Lösungsgerade für diese constraints.

5.3.3 Die Prüfung der OR–constraints

Die resultierenden Geraden b_{min} und b_{max} des Algorithmus **Check_AND_constraints** im Abschnitt 5.3.2 werden im folgenden Algorithmus **Check_AND/OR_constraints** als Anfangswerte übernommen. Es wird wieder die ursprüngliche AND/OR–constraint–Kette verwendet.

Folgende Strategie wird verfolgt:
Sei c ein OR–constraint der Sorte floor. Die Gerade b_{min} verbindet einen floor–constraint mit einem ceiling–constraint, wobei der floor–constraint in der AND/OR–constraint–Kette weiter links stehen muß als der ceiling–constraint. Das bedeutet, daß c nur zusammen mit einem ceiling–constraint rechts in der Kette eine Gerade ergeben kann, deren Steigung größer ist als die von b_{min}. Dementsprechend werden im Algorithmus alle Knoten links von c beim Vergleich der Steigungen mit b_{min} nicht betrachtet.

Genauso kann c nur zusammen mit einem ceiling–constraint in der Kette links von c eine Gerade ergeben, deren Steigung geringer ist als die von b_{max}. Es werden also im Algorithmus alle Knoten rechts von c beim Vergleich der Steigungen mit b_{max} außer acht gelasssen.

Ist c ein ceiling–constraint, verläuft die Einschränkung der in Frage kommenden Knoten ähnlich. Es bleiben beim Vergleich der Steigungen mit b_{min} alle Knoten in der Kette rechts von c unberücksichtigt und beim Vergleich der Steigungen mit b_{max} alle Knoten links von c.

Jeder Knoten erhält nun ein Attribut, das die Werte ACTIVE und INACTIVE annehmen kann. Es bestimmt, ob der jeweilige constraint im Algorithmus berücksichtigt wird oder nicht.

Die Attribute der AND–constraints sind immer auf ACTIVE gesetzt, weil sie in jedem Versuch erfüllt sein müssen. Die OR–constraints sind zu Beginn des Algorithmus alle auf INACTIVE gesetzt. Es kann in einer OR–constraint–Gruppe immer nur ein constraint ACTIVE sein, denn es genügt, daß ein OR–constraint der Gruppe erfüllt ist.

Die OR–constraint–Gruppen in der Kette werden nun von links nach rechts durchgegangen. In einer OR–constraint–Gruppe wird ein OR–constraint ACTIVE gesetzt und die untenstehende Prüfung des OR–constraint durchgeführt. Ist eine Lösungsgerade dann noch möglich, bleibt der OR–constraint ACTIVE, und der Algorithmus schreitet zur nächsten OR–constraint–Gruppe fort. Ist eine Lösungsgerade nicht mehr möglich, wird der geprüfte OR–constraint auf INACTIVE zurückgesetzt und der zweite OR–constraint dieser Gruppe auf ACTIVE gesetzt. Wenn nach der Prüfung eine Lösungsgerade möglich ist, wird die nächste OR–constraint–Gruppe betrachtet; sonst wird Backtracking ausgeführt.

Ist kein Weg durch die Kette möglich, sodaß $k_{bmin} \leq k_{bmax}$, wird der Prozeß abgebrochen. Sind dagegen nach der Prüfung der letzten OR–constraint–Gruppe b_{min} und b_{max} mit $k_{bmin} \leq k_{bmax}$

gefunden, existiert eine Lösungsgerade. Hier der Algorithmus **Check_AND/OR_constraints**.

Check_AND/OR_constraints (AND/OR–constraint–Kette)

- Alle AND–constraints werden auf ACTIVE gesetzt.
- Es werden alle möglichen Wege durch den Graphen betrachtet:
  repeat
  Der durch die Reihenfolge des Backtracking festgelegte nächste OR–constraint r wird auf ACTIVE gesetzt, falls es noch einen solchen gibt. Folgendes wird für r durchgeführt:
  - Ist r ein floor–constraint und c dessen floor–Punkt, dann wird
    - folgendes für alle Knoten s von ganz links bis exkl. r durchgeführt:
      - Ist s ein ceiling–constraint und d dessen ceiling–Punkt, dann wird die Steigung k_{bmax} mit der Steigung k der Gerade durch d und c verglichen. Ist $k < k_{bmax}$, dann b_{max}:= die Gerade durch d und c.
        Ist nun $k_{bmin} \leq k_{bmax}$, dann ist eine Lösungsgerade noch möglich und r bleibt ACTIVE; sonst wird r auf INACTIVE gesetzt.
    - folgendes für alle Knoten s von ganz rechts bis exkl. r durchgeführt:
      - Ist s ein ceiling–constraint und d dessen ceiling–Punkt, dann wird die Steigung k_{bmin} mit der Steigung k der Gerade durch c und d verglichen. Ist $k > k_{bmin}$, dann b_{min}:= die Gerade durch c und d.
        Ist nun $k_{bmin} \leq k_{bmax}$, dann ist eine Lösungsgerade noch möglich und r bleibt ACTIVE; sonst wird r auf INACTIVE gesetzt.
  - Ist r ein ceiling–constraint und c dessen ceiling–Punkt, dann wird
    - Folgendes für alle Knoten s von ganz links bis exkl. r durchgeführt:
      - Ist s ein floor–constraint und d dessen floor–Punkt, dann wird die Steigung k_{bmin} mit der Steigung k der Gerade durch d und c verglichen. Ist $k > k_{bmin}$, dann b_{min}:= die Gerade durch d und c.
        Ist nun $k_{bmin} \leq k_{bmax}$, dann ist eine Lösungsgerade noch möglich und r bleibt ACTIVE; sonst wird r auf INACTIVE gesetzt.
    - folgendes für alle Knoten s von ganz rechts bis exkl. r durchgeführt:
      - Ist s ein ceiling–constraint und d dessen ceiling–Punkt, dann wird die Steigung k_{bmax} mit der Steigung k der Gerade durch c und d verglichen. Ist $k < k_{bmax}$, dann b_{max}:= die Gerade durch c und d.
        Ist nun $k_{bmin} \leq k_{bmax}$, dann ist eine Lösungsgerade noch möglich und r bleibt ACTIVE; sonst wird r auf INACTIVE gesetzt.

  until es ist ein Weg durch den Graph gefunden und $k_{bmin} \leq k_{bmax}$ oder es wurden bereits alle Wege geprüft.

Sind b_{min} und b_{max} mit $k_{bmin} \leq k_{bmax}$ gefunden, dann gibt es mindestens eine Lösungsgerade, die diejenigen relevanten constraints erfüllt, aus denen die AND/OR–constraint–Kette gebildet wurde.

Die Lösungsgeraden, die von MatchRatio gefunden werden, sind b_{min} und b_{max} sowie sämtliche Geraden g mit $k_{bmin} < k_g < k_{bmax}$, auf denen der Schnittpunkt von b_{min} und b_{max} liegt.

5.4 Von b_{min} und b_{max} zur gesuchten Transformation

Als letzter Schritt zur Auffindung einer Transformation $w = cx+d$ müssen nun deren Parameter ermittelt werden. Wegen der Entsprechung zur Geradengleichung ergibt sich für c das Intervall $[k_{bmin}, k_{bmax}]$. Man setzt z.B.

$$c := \frac{k_{bmin} + k_{bmax}}{2}$$

Nun wird noch ein Punkt benötigt, um die Gerade zu definieren, die w bestimmt. Es könnte hier der Schnittpunkt von b_{min} und b_{max} verwendet werden. Bei der Berechnung des Schnittpunktes zweier Geraden können jedoch numerische Ungenauigkeiten das Ergebnis stören. Deshalb wird der Parameter d der gesuchten Transformation aus den constraints ermittelt.

- Ein floor–constraint begrenzt den Wert d von oben. Der zugehörige floor–Punkt wird in die Geradengleichung eingesetzt. Es muß gelten

$$cx_{floor} + d \leq y_{floor} \tag{5.2}$$
$$d \leq y_{floor} - cx_{floor} \tag{5.3}$$

- Ein ceiling–constraint begrenzt den Wert d von unten. Der zugehörige ceiling–Punkt wird in die Geradengleichung eingesetzt. Es muß gelten

$$cx_{ceil} + d \geq y_{ceil} \tag{5.4}$$
$$d \geq y_{ceil} - cx_{ceil} \tag{5.5}$$

Wurden die Ungleichungen aus allen relevanten constraints betrachtet, ergibt sich ein Intervall für d, aus dem d gewählt werden kann. Durch die Betrachtung der Grenzconstraints ist sichergestellt, daß die Lösungsgerade mit einem d aus obigem Intervall innerhalb der Beschränkungsgeraden upper und lower bleibt.

Damit ist eine Transformation $w = cx + d$ gefunden, deren zugehörige diskrete Transformation w' die Pixelzeile P auf den Bereich G ihrer selbst abbildet.

5.5 Zusammenfassung der neu eingeführten Begriffe

Dies ist eine kurze Zusammenfassung der im Kapitel 5 neu eingeführten Begriffe in alphabetischer Reihenfolge. Symbole sind hinter den Buchstaben angeführt.

Begriff	Kurze Erklärung
begin–vline	Eine vline, die durch den Mittelpunkt des ersten Pixels eines Dashs verläuft.
b_{max}	Die flachste Gerade, die einen ceiling–Punkt links mit einem floor–Punkt rechts verbindet.
b_{min}	Die steilste Gerade, die einen floor–Punkt links mit einem ceiling–Punkt rechts verbindet.
bottom	Der Index des Pixels in G, das von dem auf G projizierten bottom–Punkt einer vline getroffen wird.
bottom–Punkt	Ein Punkt in der graphischen Darstellung des Problems. Es ist der in y–Richtung kleinste Wert, den der transformierte Mittelpunkt eines Pixels innerhalb upper und lower erreichen kann.
ceiling–constraint	Ein constraint, dessen Wertebereich bei y_{top} endet. Auch kurz als ceiling bezeichnet.
ceiling–Punkt	Der Beginn eines ceiling–constraints.
constraint	Der eingeschränkte Wertebereich des Schnittpunktes der möglichen Lösungsgeraden mit einer vline.
d_{max}	Die Gerade mit der größten Steigung, die im Abschnitt 5.1.6 ein Zielpixel weiß läßt.
d_{min}	Die Gerade mit der geringsten Steigung, die im Abschnitt 5.1.6 ein Zielpixel weiß läßt.
end–vline	Eine vline, die durch den Mittelpunkt des letzten Pixels eines Dashs verläuft.
ϵ	Ein Sicherheitsabstand zur Vermeidung der Folgen numerischer Ungenauigkeiten.
floor–constraint	Ein constraint, dessen Wertebereich bei y_{bottom} beginnt. Auch kurz als floor bezeichnet.
floor–Punkt	Das Ende eines floor–constraints.
f_{max}	Die Gerade mit der größten Steigung innerhalb upper und lower.

Begriff	Kurze Erklärung
f_{min}	Die Gerade mit der geringsten Steigung innerhalb upper und lower.
G	Das Pixelmuster G ist ein Teil des Pixelmusters P. Die gesuchte Transformation soll dieses Muster ergeben.
Grenzconstraints	Die explizit eingeführten constraints an Anfang und Ende von P.
left–vline	Die linke der beiden vlines, die, falls vorhanden, für ein Zielpixel relevant sind.
lower	Eine Gerade, die wegen der erzwungenen Transformation des ersten Pixels von P auf das erste Pixel von G sowie des letzten Pixels von P auf das letzte Pixel von G die möglichen Lösungsgeraden nach unten beschränkt.
Lösungsgerade	Eine Gerade, die in der graphischen Darstellung des Problems von Abschnitt 5.1.1 der gesuchten Transformation w entspricht.
P	Das Pixelmuster P, das als Eingabe für die gesuchte Transformation dient (Abschnitt 5.1.1).
right–vline	Die rechte der beiden vlines, die, falls vorhanden, für ein Zielpixel relevant sind.
top	Der Index des Pixels in G, das von dem auf G projizierten top–Punkt einer vline getroffen wird.
top–Punkt	Ein Punkt in der graphischen Darstellung des Problems. Es ist der in y–Richtung größte Wert, den der transformierte Mittelpunkt eines Pixels innerhalb upper und lower erreichen kann.
upper	Eine Gerade, die wegen der erzwungenen Transformation des ersten Pixels von P auf das erste Pixel von G sowie des letzten Pixels von P auf das letzte Pixel von G die möglichen Lösungsgeraden nach oben beschränkt.
vline	Eine vertikale Linie, die durch den Mittelpunkt eines Pixels in P gelegt wird.
WHITE etc.	Kategorisierung der Zielpixel in G nach Vorhandensein oder Nichtvorhandensein von right–vlines und left–vlines.
x_{bottom}, y_{bottom}	Koordinaten eines bottom–Punktes
x_{ceil}, y_{ceil}	Koordinaten eines ceiling–Punktes
x_{floor}, y_{floor}	Koordinaten eines floor–Punktes
x_{top}, y_{top}	Koordinaten eines top–Punktes

Begriff	Kurze Erklärung
•	Bezeichnung für ceiling–Punkte in Abbildungen.
×	Bezeichnung für floor–Punkte in Abbildungen.

Kapitel 6

Zusammenfassung und Schluß

In dieser Arbeit wurde zunächst eine anschauliche Erklärung von IFS und dem inversen Problem gegeben. Es folgte ein Überblick der bestehenden Ansätze zur Lösung des inversen Problems. Dann wurden die entsprechenden mathematischen Grundlagen angeführt. Da der Schwerpunkt dieser Arbeit auf IFS in $\mathbb{R}$ liegt, wurden affine kontraktive Transformationen in $\mathbb{R}$ näher betrachtet. Die Berechnung ihres Fixpunktes sowie des Kontraktionsfaktors wurde gezeigt und eine alternative Darstellungsform von kontraktiven affinen Transformationen beschrieben.

Dann wurden kurz und unter Angabe von Referenzen die Diskretisierung der Transformationen von IFS und diskrete Attraktoren beschrieben, die die Basis für den folgenden Algorithmus darstellen.

Der Algorithmus MatchRatio löst das eindimensionale inverse Problem. Er liefert zu jeder beliebigen Pixelzeile ein IFS, sodaß die Pixelzeile ein diskreter Attraktor des IFS ist. Es wurde die Verwendung einer unter den Transformationen eines IFS weitgehend invarianten Eigenschaft der Pixelzeile gezeigt. Diese ist das Verhältnis der Länge von schwarzen und weißen Zusammenhangskomponenten. Da beim Abtasten die Invarianz gestört werden kann, sind die resultierenden Transformationen keine sicheren Ergebnisse, sondern mögliche Transformationen, die nur Zwischenergebnisse darstellen.

Weiters wurde beschrieben, wie aus jeder dieser möglichen Transformationen auf geometrischem Wege eine konkrete Transformation ermittelt werden kann, sofern eine solche existiert. Es wurde bewiesen, daß der dabei verwendete Lösungsweg zum korrekten Ergebnis führt. Außerdem wurden die Ergebnisse beschrieben, die MatchRatio in der Praxis liefert, und einige Eigenschaften von MatchRatio diskutiert.

MatchRatio stellt mit der Lösung des inversen Problems in $\mathbb{R}$ einen Schritt zur möglichen Lösung des inversen Problems in $\mathbb{R}^2$ dar. Das Prinzip der unter den Transformationen eines IFS invarianten Eigenschaften einer binären Pixelzeile könnte auf binäre Bilder erweitert werden. Mögliche invariante Eigenschaften in $\mathbb{R}^2$ wären Winkel zwischen Kanten, das Verhältnis der Länge von Kanten etc. Eine Fortsetzung dieser Arbeit könnte darin bestehen, einen Weg zur direkten Ermittlung von Transformationen eines IFS in $\mathbb{R}^2$ aus den obengenannten geometrischen Eigenschaften eines Bildes zu finden.

Literaturverzeichnis

[1] The Association for Computing Machinery. *SIGGRAPH '91 Course Notes—Fractal Modeling in 3-D Computer Graphics and Imaging*, volume C14, 1991.

[2] M. F. Barnsley. *Fractals Everywhere*. Academic Press, 1988.

[3] M. F. Barnsley and S. Demko. Iterated function systems and the global construction of fractals. *Proceedings of the Royal Society of London Ser. A*, 399:243–275, 1985.

[4] M. F. Barnsley, A. Jacquin, F. Malassenet, L. Reuter, and A. D. Sloan. Harnessing chaos for image synthesis. *Computer Graphics*, 22(4):131–140, Aug. 1988.

[5] M. F. Barnsley and A. D. Sloan. A better way to compress images. *BYTE*, pages 215–223, Jan. 1988.

[6] L. Berger, J.-P. Mariot, and C. Launay. A new formulation for fast image coding using quadtree representation. *Pattern Recognition Letters*, 13:425–432, June 1992.

[7] K. Falconer. *Fractal Geometry, Mathematical Foundations and Applications.* Wiley & Sons, 1990.

[8] E. Gröller. Fractals and solid modeling. *EUROGRAPHICS '92*, 11(3):C–415–C–424, 1992.

[9] A. Habibi and P. A. Wintz. Image coding by linear transformation and block quantization. *IEEE Transactions on Communications*, COM-19(1):50–63, Feb. 1971.

[10] J. C. Hart. Linear fractals in 3-D computer graphics. In *SIGGRAPH '91 Course Notes* [1], pages 145–163.

[11] J. C. Hart and T. A. DeFanti. Efficient antialiased rendering of 3-d linear fractals. *Computer Graphics*, 25(4):91–100, July 1991.

[12] A. N. Horn. IFSs and interactive image synthesis. *Computer Graphics Forum*, 9:127–137, 1990.

[13] J. E. Hutchinson. Fractals and self similarity. *Indiana University Mathematics Journal*, 30(5):713–747, 1981.

[14] A. E. Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Transactions on Image Processing*, IP-1(1):18–30, 1992.

[15] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[16] W. G. Kropatsch, M. A. Neuhauser, I. J. Leitgeb, and H. Bischof. Combining pyramidal and fractal image coding. In *Proceedings of the 11th ICPR*, volume III, pages 61–64. IAPR, IEEE Computer Society Press, Aug. 1992.

[17] J. Lévy-Véhel and A. Gagalowicz. Shape approximation by a fractal model. In G. Maréchal, editor, *Proceedings of the EUROGRAPHICS 1987*, pages 159–179. Eurographics Association, Elsevier Science Publishers, Aug. 1987.

[18] J. Lévy-Véhel and A. Gagalowicz. Fractal approximation of 2-D object. In D. A. Duce and P. Jancene, editors, *Proceedings of the EUROGRAPHICS 1988*, pages 297–311. Eurographics Association, Elsevier Science Publishers, Sept. 1988.

[19] R. Libeskind-Hadas and P. Maragos. Application of iterated function systems and skeletonization to synthesis of fractal images. In *SPIE*, volume 845 of *Visual Communications and Image Processing II*, pages 276–284, 1987.

[20] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-11(7):674–693, July 1989.

[21] B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freeman and Co., San Francisco, 1982.

[22] P. Maragos and R. Schafer. Morphological skeleton representation and coding of binary images. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-34:1228–1244, 1986.

[23] D. S. Mazel and M. H. Hayes. Using iterated function systems to model discrete sequences. *IEEE Transactions on Signal Processing*, SP-40(7):1724–1734, 1992.

[24] H. Mitsch. *Lineare Algebra und Geometrie.* Prugg Verlag, 1978.

[25] M. A. Neuhauser. Diskrete Iterierte Funktionensysteme. Diplomathesis, Technical University of Vienna, Karlsplatz 13, A–1040 Vienna, Austria, Nov. 1992.

[26] H.-O. Peitgen, H. Jürgens, and D. Saupe. *Fractals for the Classroom.* Springer Verlag, 1991.

[27] A. Pentland and B. Horowitz. A practical approach to fractal-based image compression. In *Data Compression Conference*, pages 176–185. IEEE, IEEE Computer Society Press, 1991.

[28] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, 1990.

[29] J. Stark. Iterated function systems as neural networks. *Neural Networks*, 4:679–690, 1991.

[30] E. R. Vrscay. Moment and collage methods for the inverse problem of fractal construction with iterated function systems. In *SIGGRAPH '91 Course Notes* [1], pages 271–289.