Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 81 (1990)

Heft: 9

Artikel: Datenkompression mittels iterierter Funktionensysteme (IFS) : fraktale Datenkompression, Teil 2

Autor: Gipser, Thilo

DOI: https://doi.org/10.5169/seals-903115

Datenkompression mittels iterierter Funktionensysteme (IFS)

Fraktale Datenkompression, Teil 2

Thilo Gipser

Im ersten Teil wurden Wesen und Eigenschaften von Fraktalen näher erläutert und ein Einblick in die revolutionären Auswirkungen der Fraktaltheorie auf die Wissenschaft vermittelt. Dieser zweite Teil beschäftigt sich nun mit der fraktalen Datenkompression selbst und zeigt deren Bedeutung und Möglichkeiten auf.

La première partie a expliqué la nature et les propriétés des fractales et donné une vue des effets révolutionnaires de la théorie fractale dans pratiquement tous les domaines scientifiques. Cette deuxième partie présente la compression fractale des données, son importance et ses possibilités.

Adresse des Autors

Thilo Gipser, Dipl. El.-Ing. ETH, Institut für Kommunikationstechnik, ETH-Zentrum, 8092 Zürich

Bildverarbeitung ist noch heute ein den Personal-Computern im allgemeinen vorenthaltenes Gebiet. Der Grund hierfür liegt in der enormen Datenmenge, welche die Digitalisierung von Bildern mit sich bringt und im Fehlen geeigneter Kompressionsmechanismen, um den erforderlichen Speicherplatz drastisch zu reduzieren. Man betrachte zum Beispiel eine hochauflösende Luftaufnahme, welche auf 1 m[a] vergrössert wurde und nun noch eine Auflösung von 100 Pixel/cm bei 8 signifikanten Bit pro Pixel besitzt. Dann ergibt sich eine Datenmenge von 100 MByte – viel zu viel für eine Verarbeitung auf Personal-Computern. Die gängigen Kompressionsmethoden erreichen Kompressionsfaktoren zwischen 2:1 und 10:1, die Datenmenge der Luftaufnahme liegt dann aber immer noch zwischen 10 und 50 MByte. Die fraktale Datenkompression verspricht dagegen sensationelle Datenkompressionsfaktoren von 10 000:1 und mehr!

Begründer der fraktalen Datenkompression ist Michael F. Barnsley, Mathematiker am Georgia Institute of Technology in Atlanta, USA. Seine Forschungsarbeiten erregen weltweit grosses Aufsehen; selbst die amerikanische Armee, der amerikanische Geheimdienst CIA und die Weltraumbehörde Nasa interessieren sich dafür und gewähren mittlerweile grössere finanzielle Unterstützung.

Worin besteht nun die Idee von Barnsleys fraktaler Datenkompression? Das Grundprinzip ist denkbar einfach. Wenn es gelänge, die Struktur eines gegebenen Bildes mittels eines mathematischen Ausdrucks zu erfassen, das Bild quasi in einer zugehörigen Formel aufzuschlüsseln, dann müsste anstelle des gesamten Bildes lediglich diese Formel übertragen oder abgespeichert werden. Es ist dabei absolut einleuchtend, dass die Datenmenge dieser Formel um ein Vielfaches geringer ausfällt als diejenige des gesamten Bildes; offenbar sind gewaltige Datenkompressionsfaktoren zu erwarten. Bei der Dekodierung wird das Bild einfach wieder aus dem mathematischen Ausdruck zurückentwikkelt (Bild 1).

Mit der auf dieser Idee basierenden Methode wurden am Georgia Institute of Technology farbige Bilder mit Motiven wie Sonnenblumenfelder, Wolkenstudien, Küsten, Landschaften, Schwarzwaldszenarien, Kopf eines arktischen Wolfes, Gesicht eines bolivianischen Mädchens u.v.m. komprimiert und dabei enorme Datenkompressionsfaktoren erzielt [1].

Die Grundzüge dieser Datenkompression sollen im folgenden aufgezeigt werden.

Das Chaosspiel

Um das Grundprinzip der fraktalen Datenkompression zu verstehen, sei ein recht erstaunliches Spiel, das sogenannte Chaosspiel [3], vorgestellt. Auf einem Stück Papier markiere man 3 beliebige Punkte (Fixpunkte) und gebe ihnen die Namen Rot, Blau und Grün. Weiter stehe ein Würfel zur Verfügung, dessen Seiten ebenfalls die obigen 3 Farben aufweisen (jede Farbe erscheint auf jeweils zwei Würfelflächen). Alle 6 Seiten des Würfels kommen beim Werfen mit der gleichen Wahrscheinlichkeit vor. Nun markiere man einen vierten Punkt z_0 (den Anfangspunkt) auf dem Papier – irgendwo – und beginne das Chaosspiel. Ein erster Wurf des Würfels – und die Farbe Rot liegt oben. Auf dem Blatt Papier zeichnet man einen Punkt z_1, und zwar genau in der Mitte der Strecke von z_0 und dem Fixpunkt Rot. Ein erneutes Werfen des Würfels ergibt die Farbe Grün. Man markiere den Punkt z_2 genau in der Mitte der Strecke von z_1

IFS-Code

Der Name IFS steht für iterierte Funktionensysteme (engl. *iterated function systems*). Dabei hancelt es sich um rekursivc definierte mathematische Ausdrücke; die Zusammenfassung von mehreren dieser Funktionen nennt man einen IFS-Code. Letzterer definiert nun ein ganz bestimmtes, durch Anzahl und Aufbau der einzelnen Ausdrücke festgelegtes Bild. Zur Decodierung eines IFS-Codes werden die Funktionen in völlig zufälligen Reihenfolge hintereinander aufgerufen, wodurch man eine vollständige Konstruktion des dem IFS-Code zugrunde liegenden Bildes zu erreichen vermag.

und dem Fixpunkt Grün. In diesem Sinne geht es weiter: Beim n-ten Schritt zeichne man den Punkt z_n auf das Papier, welcher genau in der Mitte zwischem dem Punkt z_{n-1} und demjenigen Fixpunkt zu liegen kommt, dessen Name mit der Farbe der nach oben zeigenden Seite des geworfenen Würfels übereinstimmt (3 Abbildungsvorschriften). Auf diese Weise markiere man 100 000 Punkte. Nun werden die ersten 20 Punkte $z_0 \ldots z_{19}$ ausgelöscht, und man betrachte die entstandene Zeichnung. Wonach sieht die derart erhaltene Punktmenge aus? Nach einer zufälligen Zusammenballung der einzelnen Punkte um die 3 Fixpunkte herum, wie man annehmen könnte? Bei weitem nicht, das Resultat zeigt Bild 2 in Form eines Sierpiński-Dreiecks.

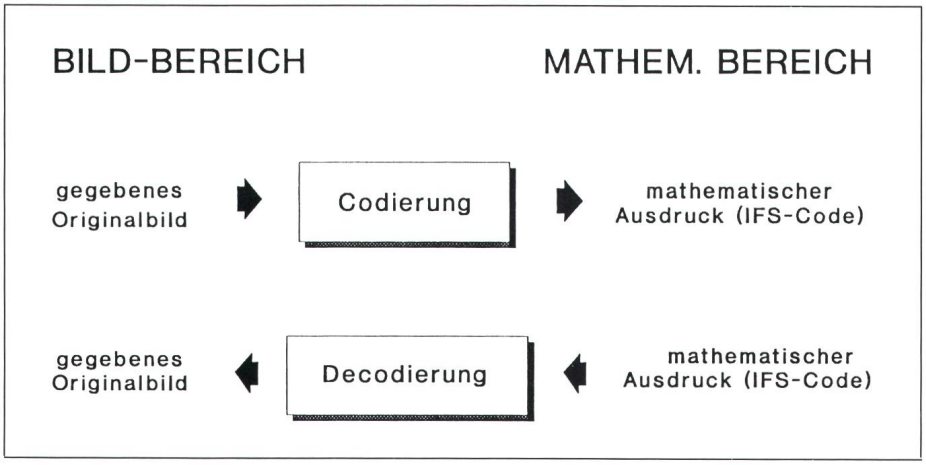


Bild 1 Funktionsprinzip der fraktalen Datenkompression

Für ein gegebenes Originalbild wird ein zugeordneter mathematischer Ausdruck bestimmt (oben). Von nun an braucht nur noch diese Formel übertragen oder abgespeichert zu werden. Um wieder das Originalbild zu erhalten, entwickelt man das Bild aus dem mathematischen Ausdruck zurück (unten).

Affine Abbildungen der fraktalen Datenkompression

Um ein beliebiges, vorgegebenes Originalbild formelmässig darstellen zu können, müssen für komplizierte Formen einfache mathematische Beschreibungen gefunden werden. Wie im ersten Beitrag dieser Publikation ausführlich dargelegt, ist dies bei fraktalen Strukturen möglich. Es liegt daher nahe, fraktale Beschreibungsmethoden zur Kodierung von Bildern heranzuziehen. Da es sich dabei bekanntlich um Rekursionsprozesse handelt, müssen anpassungsfähige, rekursiv definierte mathematische Ausdrücke Grundlage einer allgemeinen fraktalen Datenkompression sein. Solche Terme sind in den affinen Abbildungen der iterierten Funktionensysteme (IFS) gegeben.

Eine *affine Abbildung*[1] bildet - geometrisch betrachtet - eine Punktmenge durch eine beliebige Kombination aus Drehung, Streckung und Verschiebung auf eine andere Punktmenge ab (Bild 3). Hauptmerkmal einer derartigen Transformation stellt die Eigenschaft dar, dass eine gegebene Figur nicht bis zur Unkenntlichkeit entstellt, sondern ein der Ausgangsfigur ähnlich erscheinendes Abbild erzeugt wird. Grundeigenschaften der ursprünglichen Figur bleiben so erhalten (ein Dreieck bleibt ein Dreieck). Bei der affinen Transformation einer Figur wird im Prinzip jeder ihrer Punkte durch die Abbildungsvorschrift geschleust.

Eine affine Transformation w in kartesischen Koordinaten kann in allgemeiner Form wie folgt angeschrieben werden:

$$w\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \quad (1)$$

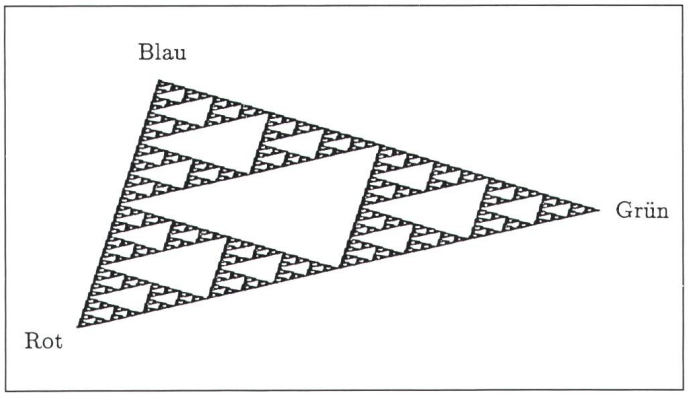
Dabei stellen die Koeffizienten a, b, c, d, e, f reelle Konstanten dar, welche entsprechend der gewünschten Transformation festgelegt werden. Um zum Beispiel die in Bild 3 angegebene Abbildung zu beschreiben, müssen obige Koeffizienten folgende Werte annehmen:

$$w\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1{,}0 & 0{,}2 \\ 0{,}3 & 0{,}8 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1{,}0 \\ 1{,}2 \end{bmatrix} \quad (2)$$

Ausgeschrieben ergibt sich der Ausdruck $w\,(x_1, x_2) = (x_1 + 0{,}2 \cdot x_2 + 1{,}0;\ 0{,}3 \cdot x_1 + 0{,}8 \cdot x_2 + 1{,}2)$, wodurch das rechtwinklige Dreieck in das nichtrechtwinklige abgebildet wird.

[1] *affin* verwandt, ähnlich

Bild 2 Chaos-Spiel



Ein klar begrenztes Sierpiński-Dreieck erscheint als Resultat des Chaos-Spiels.

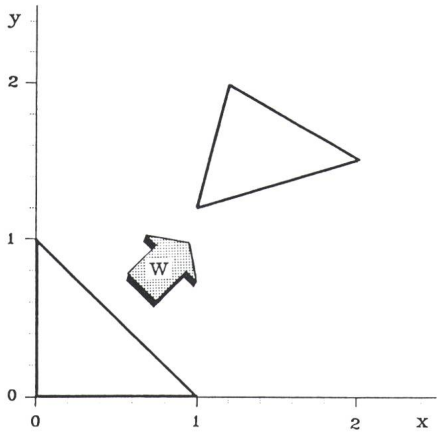


Bild 3 Affine Transformation *w*

Affin transformierte Bilder werden nicht bis zur Unkenntlichkeit verzerrt, sondern sehen dem Originalbild immer noch ähnlich.

Wie lassen sich nun die Koeffizienten einer gewünschten affinen Transformation bestimmen? Dies ist relativ einfach zu bewerkstelligen, braucht man doch nur 3 Punkte des Originalbildes (z.B. die 3 Ecken des rechtwinkligen Dreiecks in Bild 3) und die entsprechenden 3 Bildpunkte (die 3 Ekken des nichtrechtwinkligen Dreiecks) in die Gleichung (1) einzusetzen; auf diese Weise ergeben sich 6 Gleichungen, aus welchen die 6 Unbekannten a, b, c, d, e, f ermittelt werden können.

Der Vollständigkeit halber sei noch erwähnt, dass zur Beschreibung affiner Abbildungen noch andere, aussagekräftigere (äquivalente) Darstellungen als die in (1) gegebene allgemeine Form zur Verfügung stehen [4].

Fixpunkt einer affinen Abbildung

Damit eine affine Abbildung zur fraktalen Datenkompression herangezogen werden kann, muss sie *kontraktiv* sein. Eine kontraktive affine Abbildung w erfüllt die Bedingung

$$\| w(x) - w(y) \| < \| x - y \|, \qquad (3)$$

d.h. die durch die Transformation w abgebildeten Punkte $w(x)$ und $w(y)$ liegen näher beisammen als die ursprünglichen Punkte x und y. Wäre dem nicht so, lägen also die Bildpunkte weiter auseinander als die ursprünglichen Punkte, so würden mit der fortgesetzten Abarbeitung einer Transformation die Bildpunkte immer weiter (unendlich weit) voneinander wegstreben. Das Ziel ist jedoch genau das Gegenteil; die Punkte sollen später einen exakt begrenzten Bereich ausfüllen. Bei der fortgesetzten Abarbeitung einer kontraktiven Abbildung w werden die Abstände der Bildpunkte somit immer kleiner, sie streben auf einen Punkt, den *Fixpunkt* der kontraktiven Abbildung, zu. Im Chaosspiel liegen die 3 Fixpunkte der 3 Abbildungsvorschriften in den Ecken des Sierpiński-Dreiecks.

IFS-Codes

Ordnet man schliesslich jeder kontraktiven affinen Transformation w_i eine Wahrscheinlichkeit p_i zu und fasst mehrere derartige Transformationen zusammen, so erhält man den *IFS-Code* eines Bildes. Die Wahrscheinlichkeit p_i gibt dabei an, wie oft die Abbildung w_i im Vergleich zu den anderen Transformationen des IFS-Codes abgearbeitet (zufällig aufgerufen) wird; die Summe aller Wahrscheinlichkeiten p_i eines IFS-Codes hat natürlich immer den Wert 1.

Jeder IFS-Code definiert nun einen *Attraktor* (Zielbild). Der Attraktor entspricht dem bei der Dekodierung eines IFS-Codes entstehenden Bild. Im Chaosspiel wird der Attraktor durch die 3 Abbildungsvorschriften zur Ermittlung des jeweils nächsten Bildpunktes definiert und durch das entstandene Sierpiński-Dreieck dargestellt.

Die den Transformationen zugewiesenen Wahrscheinlichkeiten nehmen grundsätzlich keinen Einfluss auf Aussehen und Gestalt des Attraktors. Sie legen aber fest, welche Transformationen im Unterschied zu den übrigen wie oft abzuarbeiten sind. Dadurch können Abbildungen, die für das Zeichnen von grösseren Bildausschnitten verantwortlich sind, stärker gewichtet oder durch Hervorhebung einzelner Transformationen Schattierungen im Zielbild bewirkt werden.

Dekodierung von IFS-Codes

Die Dekodierung eines IFS-Codes ist gleichbedeutend mit der Ermittlung des einem IFS-Code zugeordneten Bildes (Attraktor). Dabei entspricht die Vorgehensweise grundsätzlich derjenigen im vorgestellten Chaosspiel.

Ein beliebiger Anfangspunkt P_0 werde festgelegt. Mittels eines Zufallszahlengenerators ist nun zufällig eine Transformation w_i des gegebenen IFS-Codes zu bestimmen und der Punkt P_0 auf den Punkt $P_1 = w_i(P_0)$ abzubilden. Im nächsten Schritt wird P_1 seinerseits durch eine weitere, erneut zufällig ausgewählte Transformation w_j in den Punkt $P_2 = w_j(P_1)$ transformiert. Nach genügend vielen Wiederholungen dieses iterativen Prozesses entspricht die entstandene Punktmenge dem Attraktor des IFS-Codes. Aus der formelmässigen Darstellung (im IFS-Code) lässt sich so das gewünschte Bild ermitteln (Bild 4).

Liegt der (frei wählbare) Anfangspunkt P_0 ausserhalb des Zielbildes, so konvergieren die Bildpunkte P_i rasch in den zugehörigen Attraktor; die ersten 10 bis 20 Punkte gehören dann aber noch nicht dazu und müssen entfernt werden. Aus diesem Grund werden letztere im allgemeinen erst gar nicht gezeichnet.

Es ist zu beachten, dass die beim Dekodierungsprozess auftretende zufälli-

Chaos

Unter Chaos versteht man ungeordnetes, nicht deterministisches Verhalten. Die Auswirkungen kleinster Veränderungen (z.B. Bahnabweichungen) bleiben nicht gering, sondern wachsen überraschend schnell an, wie z.B. beim Butterfly-Effekt. Der Ausdruck Butterfly-Effekt wurde 1963 vom amerikanischen Meteorologen E.N. Lorentz am Massachusetts Institute of Technology geprägt und verdeutlicht die Tatsache, dass unser Wetter einem gigantischen chaotischen System entspringt, in welchem bereits kleinste Einflüsse wie zum Beispiel der Flügelschlag eines Schmetterlings über das Wettergeschehen (den Hurrikan) von morgen entscheiden können. Neueste Forschungen [2] erbrachten nun die aufregende Entdeckung, dass es auch in chaotischen Systemen nicht nur völlig ungeordnet zu und her geht, sondern dass im Gegenteil eine gewisse Ordnung im Chaos vorherrschen kann. Die Zustände eines Systems bewegen sich dann auf ganz bestimmten Bahnen oder Gebieten, sog. Attraktoren (von lat. *attrahere* anziehen), sind also in gewisser Weise vorhersagbar.

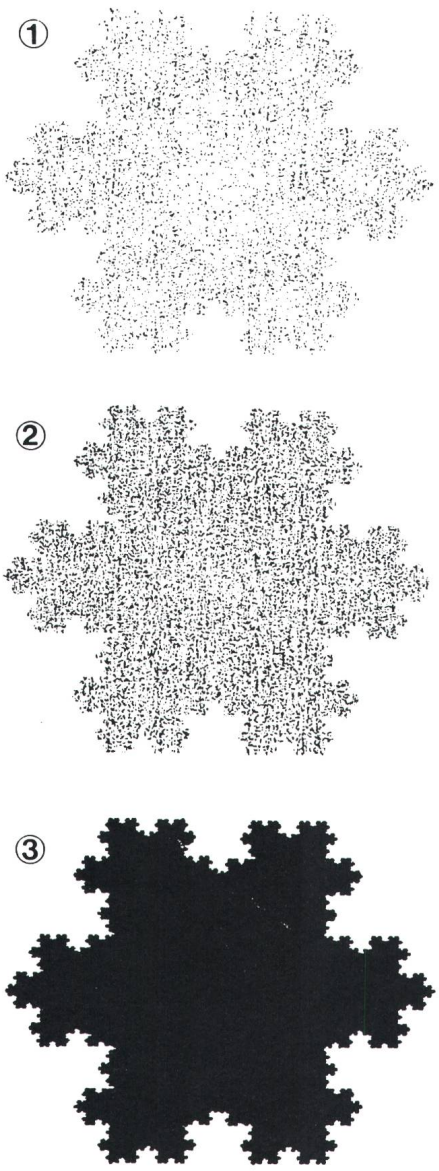


Bild 4 Decodierung eines IFS-Codes

Bei der Decodierung entsteht aus einem IFS-Code nach und nach (wie im Chaos-Spiel) das zugehörige Zielbild; in diesem Fall handelt es sich dabei um die Kochsche Schneeflockenkurve.

ge Reihenfolge der Transformationen absolut keinen Einfluss auf das entstehende Zielbild ausübt. Die zugeordneten Wahrscheinlichkeiten p_i sind jedoch dafür verantwortlich, wie oft die einzelnen Abbildungen w_i im Vergleich zu den übrigen Transformationen aufgerufen werden.

Kodierung der Originalbilder

Bislang war stets von der Dekodierung eines IFS-Codes die Rede. Es stellt sich nun umgekehrt die Frage, wie aus einem gegebenen Originalbild der zugehörige IFS-Code - also die *Formel* eines Bildes - ermittelt werden kann. Dieses inverse Problem ist ungleich schwieriger zu lösen, und seiner Bewältigung kommt fundamentale Bedeutung zu.

Die triviale Lösung bestünde darin, zu jedem Punkt eines durch ein Punktraster gegebenen Originalbildes eine affine Abbildung zu bestimmen, welche das gesamte Bild auf diesen Punkt zusammenschrumpfen lässt. Der Attraktor des so erhaltenen IFS-Codes wäre dann offensichtlich mit dem Ursprungsbild identisch. Dieses Vorgehen ist aber nicht besonders zweckmässig, da hierbei genauso viele affine Transformationen benötigt würden, wie das Originalbild Punkte hat - die resultierenden Datenkompressionsfaktoren wären also alles andere als beeindruckend. Hier hilft das sogenannte Collage-Theorem weiter, das im folgenden näher erläutert werden soll.

Es sei ein digitalisiertes Ursprungsbild T gegeben (z.B. ein schwarzes Blatt auf einem weissen Hintergrund). Eine affine Transformation w_1 werde eingeführt und das dadurch bestimmte Subbild $w_1(T)$ (welches eine verkleinerte, transformierte Kopie von T darstellt) berechnet. Dieses Bild verschiebt man nun so lange, bis es einen Teil von T möglichst gut überdeckt. Dabei muss $w_1(T)$ als Teilmenge der Punkte erscheinen, die T repräsentieren, d.h. die überlappenden Ränder von T und $w_1(T)$ sollten möglichst genau übereinstimmen, im Idealfall deckungsgleich sein. Alsdann wählt man eine zweite Transformation und verschiebt das Bild $w_2(T)$ erneut derart, dass es einen weiteren Ausschnitt von T überlagert. Bei alledem ist die Überlappung der transformierten Bilder $w_1(T)$ und $w_2(T)$ möglichst gering zu halten (Überlappungen führen nicht zu Fehldekodierungen, wohl aber zu einer Verminderung der Dekodierungsgeschwindigkeit). Auf diese Weise fährt man fort, bis schliesslich das Originalbild T vollständig von affin transformierten Kopien $w_i(T)$ überdeckt wird (Bild 5). Dieser Prozess der vollständigen Überlagerung eines Originalbildes T durch kleinere Subbilder $w_i(T)$ heisst im Englischen *Tiling*[2].

Mit diesem Vorgehen ergibt sich eine Anzahl kontraktiver affiner Abbildungen, die durch die finalen Positionen der Subbilder relativ zum Ur-

[2]engl. *tiling* das Dachdecken, das Kachellegen

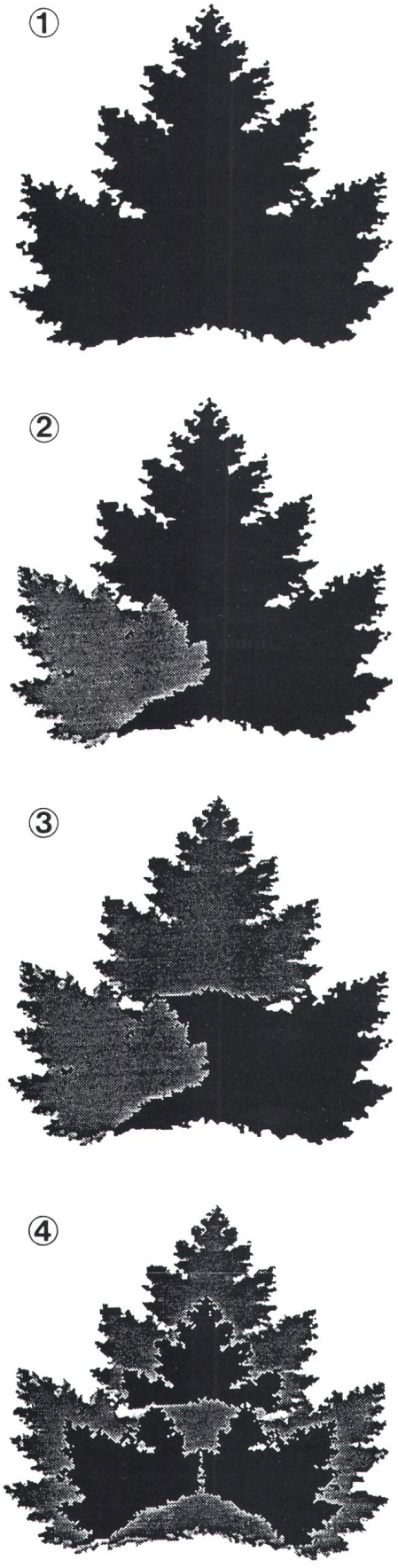


Bild 5 Codierung eines Bildes

Das Originalbild (hier ein Ahornblatt) wird sukzessive mit Subbildern überdeckt. Wählt man letztere nicht optimal aus, so entstehen Überlappungen (im Bild erkennbar an den Schattierungsänderungen).

sprungsbild bestimmt sind und über welche das *Collage-Theorem* folgenden Aussage macht: Kann ein Originalbild T annähernd vollständig mit kleineren affinen Transformationen $w_i(T)$ seiner selbst überdeckt werden, dann lässt sich eine Approximation des Originalbildes konstruieren, indem der Attraktor der Gesamtheit der affinen Abbildungen berechnet wird. Ist eine vollständige Überdeckung möglich, so ist eine exakte Rekonstruktion des Originalbildes gegeben.

Erläuterungen und Beispiele

Die Zahl der beim Kodierungsprozess unter Verwendung des Collage-Theorems erhaltenen Subbilder entspricht der Anzahl der für die Kodierung des Originalbildes erforderlichen Transformationen. Die Schwierigkeit liegt darin, einerseits das Ursprungsbild durch möglichst wenig affin transformierte Kopien zu überdecken, anderseits aber die Überlappungen der einzelnen Subbilder minimal zu halten. Es ist somit eine möglichst kleine Familie von affinen Abbildungen zu bestimmen, welche das gegebene Bild mit einer gewünschten Auflösung darstellt.

Interessant ist die folgende Anmerkung zur eben erwähnten Auflösung eines Zielbildes. Bis zu der geforderten Auflösung ist der Attraktor bei perfekter fraktaler Kodierung vom Originalbild nicht zu unterscheiden. Vergrössert man ihn aber weiter, so wird selbstverständlich ein fehlerhaftes Ergebnis erzielt: Da das Zielbild ein Fraktalbild ist (ein Attraktor wird ja rekursiv aufgebaut und hat grundsätzlich fraktalen Charakter), werden bei zunehmender Vergrösserung immer feinere Details selbst bei einem Massstab enthüllt, wo auf dem Originalbild nur noch ein einziger Bildpunkt erscheint.

Soll nun ein differenzierter aufgebautes Originalbild, zum Beispiel das Bild einer verregneten Meeresküste, kodiert werden, so lässt sich dieses immer in einfachere Teilbilder aufspalten. Die Teilbilder werden einzeln kodiert und das Zielbild dann aus ihrer Gesamtheit zusammengesetzt. Im erwähnten Beispiel lässt sich das Originalbild etwa wie folgt aufteilen: Regen, Wolken, Himmel, Felsen im Wasser in Küstennähe, Meer, Vögel in der Luft, Streifen Strand und Gras entlang dieses Strandes.

IFS-Codes weisen eine fundamentale Stabilität auf: Der IFS-Code muss nicht völlig exakt bestimmt werden, um eine gute Ähnlichkeit mit dem Originalbild zu erreichen. Er ist im Gegenteil robust, d.h. kleine Änderungen im Code führen *nicht* zu unakzeptablen Schäden im Zielbild. Dies steht im Gegensatz zu vielen anderen rekursiv definierten Algorithmen (etwa zur Erzeugung von pflanzlichen Strukturen), bei welchen das erhaltene Endprodukt von der genauen Sequenz der Zufallszahlen während der Berechnung abhängt. Diese Stabilität ist ein wichtiges Merkmal im Hinblick auf die Systemunabhängigkeit und die interaktive Bearbeitung von IFS-Codes am Computer.

Im folgenden sollen zur Illustration einige IFS-Codes angegeben werden. Sie sind in Bild 6 zusammengestellt. Interessant ist vor allem der IFS-Code des im letzten Kapitel kodierten Ahornblatts einschliesslich seines Attraktors gemäss Bild 7. Daraus ersieht man auch die Aussage des Collage-Theorems, dass eine ungenaue Überdeckung des Originalbildes durch Subbilder zu einem ungenauen Zielbild führt. Man beachte weiter die Werte der Koeffizienten beim Sierpiński-Dreieck: Wie leicht ersichtlich ist, stellen sie nichts anderes als die formelmässige Beschreibung der 3 Abbildungsvorschriften aus dem Chaosspiel dar. Die ganze Brisanz der fraktalen Datenkompression verdeutlicht das letzte Beispiel: Aus nur 4 (!) affinen Transformationen kann das Farnblatt in Bild 8 konstruiert werden.

Bild 6 Zusammenstellung der in diesem Beitrag erwähnten IFS-Codes

Objekt	w	a	b	c	d	e	f	p
Koch'sche Insel	*1*	0.34	0.0	0.0	0.33	168	47	0.13
	2	0.34	0.0	0.0	0.33	268	47	0.13
	3	0.34	0.0	0.0	0.33	318	130	0.13
	4	0.34	0.0	0.0	0.33	268	213	0.13
	5	0.34	0.0	0.0	0.33	168	213	0.13
	6	0.34	0.0	0.0	0.33	120	130	0.13
	7	0.0	0.56	-0.56	0.0	225	373	0.22
Ahorn-Blatt	*1*	0.40	0.38	-0.25	0.44	25	209	0.21
	2	0.59	0.0	-0.01	0.60	126	16	0.28
	3	0.40	-0.35	0.26	0.42	309	73	0.21
	4	0.74	0.03	-0.02	0.74	67	80	0.30
Sierpiński-Dreieck	*1*	0.5	0.0	0.0	0.5	0.0	0.0	0.33
	2	0.5	0.0	0.0	0.5	1.0	0.5	0.33
	3	0.5	0.0	0.0	0.5	0.5	0.5	0.34
Farn	*1*	0.0	0.0	0.0	0.16	0.0	0.0	0.01
	2	0.85	0.04	-0.04	0.85	0.0	1.6	0.85
	3	0.20	-0.23	0.23	0.20	0.0	1.6	0.07
	4	-0.15	0.28	0.26	0.24	0.0	0.44	0.07

Farben und dreidimensionale Darstellung

Es soll noch kurz auf die Möglichkeit eingegangen werden, farbige und dreidimensionale Bilder mittels IFS-Codes auf dem Computer zu erzeugen. Ordnet man im einfachsten Fall jeder Transformation eine eigene Farbe zu, so zeigen sich sehr schön die Subbilder, aus welchen der Attraktor zusammengesetzt ist. Eine interessantere Methode besteht darin, jeweils die Farbe zu ändern, wenn ein Punkt an der gleichen Stelle wiederholt gezeichnet wird. Dann sind die den Transformationen zugeordneten Wahrscheinlichkeiten direkt für die Farbgebung des erzeugten Zielbildes verantwortlich.

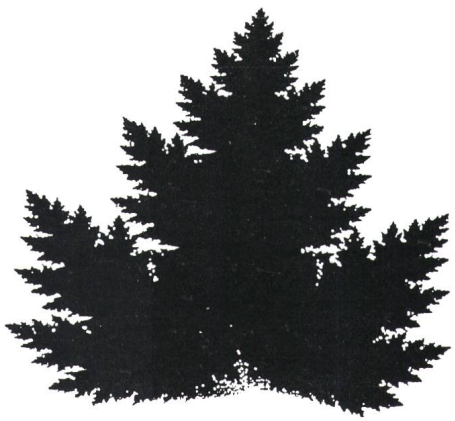


Bild 7 Attraktor des Ahornblattes
Man vergleiche die Übereinstimmung mit dem Originalbild

Bild 8 Farnblatt
Aus einem nur 4 affine Transformationen umfassenden IFS-Code lässt sich dieses Farnblatt decodieren.

Natürlich können IFS-Transformationen auch auf die dritte Dimension ausgeweitet werden. In diesem Fall ergeben sich in Gleichung (1) dreidimensionale Vektoren und eine 3×3-Abbildungsmatrix. Zur graphischen Darstellung auf dem Computer sind geeignete Algorithmen zu verwenden, die eine Projektion des nunmehr dreidimensionalen Attraktors auf den zweidimensionalen Bildschirm ermöglichen.

Fazit und Schlussbemerkungen

Anlässlich einer Diplomarbeit [5] am Institut für Kommunikationstechnik der ETH Zürich wurde eine komplette IFS-Umgebung zur Ermittlung und Dekodierung von IFS-Codes aufgebaut (IFS-Tools), welcher auch die hier präsentierten Bilder entstammen. Mit ihrer Hilfe können Bilder in den Computer eingelesen, mittels eines Collage-Editors interaktiv kodiert, die ermittelten IFS-Codes analysiert, modifiziert, abgespeichert und natürlich auch dekodiert werden. Verschiedene Algorithmen stehen für die Bestimmung der Kodierungsqualität zur Verfügung (so ist es zwecks Ermittlung der Kodierungsgenauigkeit möglich, Originalbild und Attraktor quasi übereinanderzulegen). Diverse Hilfsprogramme dienen der Vereinfachung des Umgangs mit dem Softwarepaket. Es gelang eine Kodierung der verschiedensten Originalbilder, vor allem auch die Bestimmung der IFS-Codes von vielen nichtfraktalen Strukturen wie Rechtecken, Quadraten, Dreiecken, Geraden, Kreisen, Spiralen u.a. Die Untersuchungen werden fortgesetzt.

Potentielle Anwendungsgebiete der fraktalen Datenkompression dürften in Zukunft überall dort liegen, wo digitale Bilder zu übertragen oder abzuspeichern sind. Grosses Interesse besteht ferner in all den Bereichen, welche sich mit der Archivierung von Bildmaterial auseinanderzusetzen haben (Raumfahrt, Satellitenaufklärung). Da die anfallenden Datenmengen dort häufig zu gross sind, sieht man sich infolge fehlender geeigneter Kompressionsmethoden oftmals gezwungen, beträchtliche Datenmengen zu vernichten. Ein weiterer Einsatzbereich zeichnet sich im Bereich der Modellierungs- und Simulationstechniken ab [6].

Die fraktale Datenkompression ist allerdings noch nicht ausgereift. Insbesondere reichen die Kodierungsmöglichkeiten von Bildern mit Hilfe des Collage-Theorems nicht aus, um beliebige Originalbilder rasch kodieren zu können. Hier sind neue Kodieralgorithmen zu entwickeln; denkbar wäre zum Beispiel im Hinblick auf die Stabilität der IFS-Codes eine rückgekoppelte, automatisierte Änderung der Abbildungsparameter.

Der weitere Erfolg der fraktalen Datenkompression dürfte auch eng mit dem der Entwicklung von Bilderkennungsmethoden verknüpft sein. Eine damit verbundene, anzustrebende Automatisierung der Bildkodierung, das heisst ein selbständiges Auffinden fraktaler Grundmuster unter Verwendung des Collage-Theorems oder anderer Algorithmen, würde einen grossen Schritt nach vorn bedeuten.

Keine Hindernisse sieht der Autor hingegen in der Tatsache, dass der vollständige Bildaufbau beim Dekodierungsprozess heute noch eine bis mehrere Minuten in Anspruch nimmt[3]. Einerseits weist die Computerentwicklung klar in Richtung höherer Leistungsfähigkeit und damit auch in Richtung höherer Verarbeitungsgeschwindigkeiten, anderseits kann die Dekodierung eines gegebenen IFS-Codes problemlos von mehreren Prozessoren gleichzeitig vorgenommen und die Dekodierungsgeschwindigkeit dadurch beträchtlich gesteigert werden. Weil die Reihenfolge der von den einzelnen Prozessoren aufgerufenen Abbildungsvorschriften zufällig und damit unterschiedlich ist, werden von allen Prozessen gleichzeitig verschiedene Bereiche des Zielbildes aufgebaut.

Bleibt zu erwähnen, dass es sich bei den Forschungsgebieten im Bereich der Fraktale, des deterministischen Chaos und vor allem auch der fraktalen Datenkompression um durchwegs sehr junge Disziplinen handelt, deren Bearbeitung und Nutzung erst durch die Entwicklung und den Einsatz von Computer überhaupt ermöglicht wurden. Trozu ihrer kurzen Entwichlungsphase sind bereits interessante Erkenntnisse gewonnen worden. Die fraktale Datenkompression, die noch in den Kinderschuhen steckt, dürfte in Zukunft noch zu reden geben.

[3] IBM PC PS/2,50 (CPU 80286, 10 MHz)

Literatur

[1] *M.F. Barnsley* and *A.D. Sloan:* A better way to compress images. Byte 13(1988)1, p. 215...223.

[2] *J.P. Crutchfield* a.o.: Chaos. Spektrum der Wissenschaft -(1987)2, S.78...90.

[3] *M.F. Barnsley:* Fractal modelling of real world images. In: The science of fractal images. Edited by H. Peitgen and D. Saupe. New York a.o., Springer-Verlag, 1988; p. 219...242.

[4] *T. Gipser* und *U. Künzli:* Fraktale Datenkompression. Teil 1. Semesterarbeit am Institut für Kommunikationstechnik, ETH Zürich, 15. Juli 1988.

[5] *T. Gipser* und *U. Künzli:* Fraktale Datenkompression. Teil 2. Diplomarbeit am Institut für Kommunikationstechnik, ETH Zürich, 5. Dezember 1988.

[6] *G. Zorpette:* Fractals: not just another pretty picture. IEEE Spectrum 25(1988)10, P. 29...31.