(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 698 37 430 T2** 2007.10.31

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 0 980 564 B1**
(21) Deutsches Aktenzeichen: **698 37 430.4**
(86) PCT-Aktenzeichen: **PCT/US98/09290**
(96) Europäisches Aktenzeichen: **98 921 013.3**
(87) PCT-Veröffentlichungs-Nr.: **WO 1998/050886**
(86) PCT-Anmeldetag: **06.05.1998**
(87) Veröffentlichungstag der PCT-Anmeldung: **12.11.1998**
(97) Erstveröffentlichung durch das EPA: **23.02.2000**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **28.03.2007**
(47) Veröffentlichungstag im Patentblatt: **31.10.2007**

(51) Int Cl.[8]: ***G06K 9/36*** (2006.01)
G06K 9/46 (2006.01)
G06T 9/00 (2006.01)

(30) Unionspriorität:
45915 P **07.05.1997** **US**

(84) Benannte Vertragsstaaten:
DE, FR, GB, IT, NL

(73) Patentinhaber:
Landmark Graphics Corp., Houston, Tex., US

(72) Erfinder:
HALE, Ira D., Englewood, CO 80112, US

(74) Vertreter:
HOFFMANN & EITLE, 81925 München

(54) Bezeichnung: **VERFAHREN ZUR DATENKOMPRESSION**

Beschreibung

Hintergrund der Erfindung

Gebiet der Erfindung

[0001] Die Erfindung bezieht sich allgemein auf Datenkompressionsverfahren und insbesondere auf eine Verbesserung des JPEG-Verfahrens zur Bildkompression, wie es auf seismische Daten angewandt wird.

Stand der Technik

[0002] Der JPEG-Standardalgorithmus zur Bildkompression (Pennebaker and Mitchell, 1993) besteht aus den folgenden drei Schritten, die für jeden 8 × 8-Block von Pixeln in einem zweidimensionalen Array durchgeführt werden:

1. Transformieren des 8 × 8 Blocks von Pixeln unter Verwendung einer diskreten Kosinustransformation.
2. Quantisieren (Skalieren und Abrunden) der Transformationskoeffizienten in kleine ganze Zahlen.
3. Kodieren der Bits unter Verwendung weniger Bits zur Darstellung der häufigsten ganzen Zahlen.

[0003] Der Dekompressionsalgorithmus kehrt jeden dieser Schritte in umgekehrter Reihenfolge um. Beide Algorithmen können leicht ausgedehnt werden, um Arrays jeder Dimension zu komprimieren und zu dekomprimieren.

[0004] Fig. 1 zeigt ein 2D-Array seismischer Daten, das nicht komprimiert wurde. Das 2D-Array aus 32 Bit Gleitkommazahlen der Fig. 1 ist ein aus einer 3D seismischen Vermessung extrahierter Schnitt mit konstanter Zeit. Die Fig. 2 zeigt einen vergrößerten Teilsatz desselben Arrays.

[0005] Fig. 3 zeigt denselben vergrößerten Teilsatz nach Kompression und Dekompression des gesamten 2D-Arrays unter Verwendung eines JPEG-ähnlichen Algorithmus. Das Kompressionsverhältnis für das gesamte Array beträgt ungefähr 103 : 1, was bedeutet, dass das ursprüngliche Array aus 32 Bit Gleitkommazahlen ungefähr 103 mal so viele Bits wie das komprimierte Array enthielt.

[0006] Für solche großen Kompressionsverhältnisse erzeugt dieser JPEG-ähnliche Algorithmus die in Fig. 3 sichtbaren Blockartefakte. Bei niedrigeren Kompressionsverhältnissen werden diese Unstetigkeiten zwischen den Blöcken weniger sichtbar, jedoch können sie immer noch bedeutsam sein, insbesondere, wenn eine weitere Verarbeitung oder Interpretation nach der Dekompression durchgeführt wird.

[0007] Die Artefakte in Fig. 3 sind das Ergebnis davon, dass jeder Block von 8×8 Proben unabhängig komprimiert und dekomprimiert wurde, wobei nicht versucht wurde, eine Kontinuität zwischen den Blöcken aufrechtzuerhalten.

[0008] Trotz dieser Artefakte ist die Fähigkeit, solche kleinen Teilsätze von Daten unabhängig zu komprimieren und zu dekomprimieren, ein wünschenswertes Merkmal. Insbesondere ermöglicht es Zugang zu einem kleinen Teil eines großen komprimierten Arrays, ohne dass gesamte Array dekomprimieren zu müssen. Es ermöglicht auch, dass der Kompressionsalgorithmus sich an räumliche Variationen in der Datenamplitude und im Spektrum anpasst. Diese Merkmale fehlen bei Kompressionsverfahren, die auf Wavelet-Transformationen (z. B. Bradley et al., 1993; Wickerhauser, 1994) basieren. Das in dieser Anmeldung angegangene Problem ist es, diese Merkmale ohne die Blockartefakte zu erhalten.

[0009] Eine Lösung dieses Problems ist es, die Daten unter Verwendung überlappender Blöcke zu komprimieren, so dass die dekomprimierten Probenwerte ohne Bezug auf Werte neben den Blockgrenzen berechnet werden können. Diese Lösung wurde von Yeo und Liu (1995) in ihrer Adaptation des JPEG-Algorithmus auf das Volumenrendering von medizinischen 3D-Bildern verwendet. Leider erhöht die Verwendung überlappender Blöcke die Anzahl der zu komprimierenden Blöcke, was die Rechenzeiten erhöht und die Kompressionsverhältnisse verringert.

Bezugnahmen auf in dieser Beschreibung zitierte frühere Arbeiten

Bradley, J. N., C. M., and Hopper, T., 1993, The FBI wavelet/scalar quantization standard for gray-scale fingerprint image compression: Visual Information Processing II, SPIE Proceedings, 293-304. (ftp://ftp.c3.lanl.gov/pub/WSQ).

Jawerth, B., and Sweldens, W., 1995, Biorthogonal smooth local trigonometric bases: J. Fourier Anal. Appl., 2. (http://cm.bell-labs.com/who/wim/papers/-papers.html).
Jawerth, B., Liu, Y., and Sweldens, W., 1996, Signal compression with smooth local trigonometric bases: http://cm.bell-labs.com/who/wim/papers/-papers.html
Malvar, H. S., and Staelin, D. H., 1989, The LOT-transform coding without blocking effects: IEEE Transactions on Acoustic, Speech, and Signal Processing, 37, no 4, 553-559.
Malvar, H. S., 1990, Lapped transforms for efficient transform/subband coding: IEEE Transactions on Acoustic, Speech, and Signal Processing, 38, no. 6, 969-978.
Pennebaker, W. B., and Mitchell, J. L., 1993, JPEG still image data compression standard: Van Nostrand Reinhold.
Princen, J. P., and Bradley, A. B., 1956, Analysis/synthesis filter bank design based on time domain aliasing cancellation: IEEE Transaction on Acoustics, Speech, and Signal Processing, 34, no. 5, 1153-1161.
Wickerhauser, M. V., 1994, Adapted wavelet analysis from theory to software: A. K. Peters, Wellesley, Massachusetts. (XP002196921).
Yeo, B., and Liu, B., 1995, Volume rendering of DCT-based compressed 3D scalar date: IEEE Transactions on Visualization and ComputerGraphics, 1, no. 1, 29-43.
Aharoni, G. et al, March 1993, Local cosine transform – a method for the reduction of the blocking effect in JPEG: Journal of Mathematical Imaging and Vision, 3, no. 1, 7-38 (XP000601160 ISSN: 0924-9907).

Aufgabe der Erfindung

[0010] Es ist eine Hauptaufgabe der Erfindung, einen verbesserten JPEG-Algorithmus zur Datenkompression bereitzustellen.

[0011] Eine wichtige Aufgabe der Erfindung ist es, einen verbesserten JPEG-Datenkompressionsalgorithmus bereitzustellen, welcher verwendet werden kann, um kleine Teilsätze von Daten ohne Blockartefakte zu komprimieren.

Darstellung der Erfindung

[0012] Die Erfindung besteht aus einem Verfahren und einem Computerprogramm zur Datenkompression, welches JPEG-Verfahren für diskrete Kosinustransformationen und für die Huffmann-Codierung der quantisierten Transformationskoeffizienten verwendet, und zwar gemäß den folgenden Ansprüchen. Das Verfahren ist eine Verbesserung gegenüber den Standard-JPEG-Verfahren, hauptsächlich in den zusätzlichen Schritten zur Vermeidung von Blockartefakten und in der Quantisierung der Transformationskoeffizienten.

Kurze Beschreibung der Zeichnungen

[0013] Fig. 1 stellt einen Schnitt konstanter Zeit dar, der aus einer seismischen 3D-Vermessung extrahiert wurde, wobei die Anzeigen ein 2D-Array von 32 Bit Gleitkommazahlen ist, welches nicht komprimiert wurde;

[0014] Fig. 2 ist ein vergrößerter Teilsatz des 2D-Arrays der Fig. 1;

[0015] Fig. 3 zeigt Daten der Fig. 2 nach Kompression und Dekompression mit Hilfe einer direkten Adaptation des JPEG-Algorithmus, wobei die Blockartefakte, d. h. die Unstetigkeiten zwischen den 8×8-Probenblocks, die unabhängig komprimiert wurden, gezeigt sind;

[0016] Fig. 4 zeigt die Daten der Fig. 2 nach Kompression und Dekompression mit Hilfe eines JPEG-ähnlichen Algorithmus, der konzipiert wurde, um die Blockartefakte zu unterdrücken;

[0017] Fig. 5 zeigt eine Matrix C_{III}^{-1}, die einer umgekehrten DCT-III mit Länge 32 entspricht, wobei schwarze Pixel positiven Werten entsprechen; und weiße Pixel negativen Werten entsprechen;

[0018] Fig. 6 zeigt eine Matrix, die einer umgekehrten blockierten DCT-III mit Blocklänge 8 entspricht;

[0019] Fig. 7 zeigt Spalten 11 und 19 der invertierten blockierten DCT-III Matrix der Fig. 6, wobei die Blockartefakte in der JPEG-Kompression von Unstetigkeiten verursacht sind, wie jene, die zwischen den Proben 15, 16 gezeigt sind, und die erste Probe in jedem Kosinus liegt neben der gezeichneten Kurve und in dieser Figur aufgrund der Unstetigkeit in der Funktion b(j), die von der folgenden Gleichung (1c) definiert ist;

[0020] Fig. 8 zeigt geglättete zusammenlaufende Kosinusse, die erhalten wurden, indem die in Fig. 7 gezeigten, begrenzten (trunkierten) Kosinusse, welche Spalten 11 und 19 der invertierten gefalteten DCT-III Matrix der Fig. 9 sind, aufgefaltet werden;

[0021] Fig. 9 zeigt eine Matrix, die einer invertierten gefalteten DCT-III entspricht, wobei die Spalten dieser Matrix sich von Block zu Block überlappen, und zwar mit Werten, die gleichmäßig auf Null zulaufen, so dass die Gewichte der Summe dieser Spalten keine Blockartefakte aufweisen.

[0022] Fig. 10 zeigt eine Matrix, die der Operation entspricht, die verwendet wird, um die invertierte blockierte DCT-III aufzufalten, wobei die Matrix in Fig. 9 gleich dem Produkt dieser Matrix und jener in Fig. 6 dargestellten ist;

[0023] Fig. 11 zeigt eine Auffaltoperation, welche eine paarweise Kombinierung und ein Austausch der Probenwerte über die DCT-Blockgrenzen hinaus ist, und wobei das Falten das Inverse dieser Operation ist;

[0024] Fig. 12 veranschaulicht die Schritte zum Dekomprimieren der hervorgehobenen Probe in Block A, was erfordert, dass man zuerst die DCT-III Blöcke A, B, C und D dekodiert, dequantisiert und invertiert und dann die vier hervorgehobenen Proben auffaltet, wobei die Falt- und Auffaltoperationen ein paarweises Mischen der Proben über die DCT-Blockgrenzen hinaus sind.

Darstellung der bevorzugten Ausführungsformen der Erfindung

[0025] Diese Erfindung besteht aus einem verbesserten Verfahren zur Datenkompression basierend auf der Vermischung von Daten in benachbarten Blöcken und während der Kompression mit einem JPEG-ähnlichen Algorithmus. Die JPEG-ähnliche Datenkompression wurde in verschiedenen Formen von vielen Autoren beschrieben (z. B. Princen and Bradley, 1986; Malvar and Stealin, 1989; Malvar, 1990; Wickerhauser, 1994; Jawerth et al., 1995). Der Vorteil der JPEG-ähnlichen Datenkompression ist jener, dass er dazu neigt, die Kompressionsverhältnisse zu erhöhen und dabei die Berechnungszeiten lediglich geringfügig erhöht.

[0026] Fig. 4 zeigt denselben Teilsatz aus dem 2D-Array der Fig. 1 nach Kompression und Dekompression unter Verwendung eines zweiten JPEG-ähnlichen Algorithmus, der auf dem unten beschriebenen Verfahren der Erfindung basiert. Das Kompressionsverhältnis für das gesamte Array beträgt ungefähr 112 : 1. Blockartefakte fehlen in diesen dekomprimierten Daten.

[0027] Der Unterschied zwischen den früheren JPEG-ähnlichen Algorithmen und dem neuen JPEG-ähnlichen Algorithmus gemäß der Erfindung liegt in einer Abwandlung der blockierten diskreten Kosinustransformation, die im JPEG-Standard spezifiziert ist. Diese Abwandlung im Algorithmus dieser Erfindung schwächt die Blockartefakte ab, während er wünschenswerte Merkmale der früheren JPEG-ähnlichen Algorithmen beibehält.

Diskrete Kosinustransformation

[0028] Die von der früheren JPEG-Kompression verwendete Transformation ist eine diskrete Kosinustransformation, die DCT-II genannt wird. Gemäß dieser Erfindung wird eine andere diskrete Kosinustransformation verwendet, die DCT-III genannt wird (für eine Aufzählung diskreter Kosinustransformationen wird auf Wickerhauser, 1994, Seite 84 verwiesen)

DCT-III

[0029] Die Vorwärtstransformation DCT-III ist definiert durch

$$z(k) = \sum_{j=0}^{M-1} y(j)b(j)\sqrt{\frac{2}{M}}\cos\left[\frac{\pi(2k+1)j}{2M}\right]$$

k = 0,1,....., M - 1 (1a)

und die entsprechende inverse Transformation ist

$$y(j) = \sum_{k=0}^{M-1} z(k)b(j)\sqrt{\frac{2}{M}}\cos\left[\frac{\pi(2k+1)j}{2M}\right]$$

j = 0,1, ..., M - 1 (1b)

wobei

$$b(j) \equiv \begin{cases} \frac{1}{\sqrt{2}}, & j=0, \\ 1, & otherwise. \end{cases}$$

[0030] Die vorwärts gerichtete und invertierte Transformation können als Matrixmultiplikationen dargestellt werden, wie in $z = C_{III}y$ und $y = C_{III}^{-1}z$, wobei die Matrix C_{III} die folgenden Elemente besitzt:

$$C_{III}(k,j) = b(j)\sqrt{\frac{2}{M}}\cos\left[\frac{\pi(2k+1)j}{2M}\right] \quad (1d)$$

[0031] Die invertierte Transformationsmatrix C_{III}^{-1} ist in Fig. 5 dargestellt, M = 32. Jeder Vektor von 32 reellen Zahlen kann als gewichtete Summe der Spalten dieser Matrix dargestellt werden.

[0032] In einem eindimensionalen Kompressionsalgorithmus kann ein Vektor y aus Probenwerten mit lediglich wenigen Spalten aus der Matrix C_{III}^{-1} angenähert werden. Die Gewichte für jede Spalte würden von den Transformationskoeffizienten im Vektor z gegeben werden. Hohe Kompressionsverhältnisse erfordern, dass viele der Koeffizienten in z vernachlässigbar und fast Null sind. Solch kleine Koeffizienten werden zu Null quantisiert, was unter Verwendung weniger Bits effizient kodiert werden kann.

[0033] Der JPEG-Standard beschreibt die Kompression von zweidimensionalen Bildern, nicht eindimensionalen Vektoren. Die in der JPEG-Kompression zuvor verwendete DCT-II ist eine 2D-Transformation. Da jedoch die diskrete Kosinustransformation mehr-dimensionaler Daten als Kaskade von 1D-Transformationen entlang jeder Datendimension durchgeführt werden kann, wird der Einfachheit halber lediglich die 1D-Transformation beschrieben.

Blockierte DCT-III

[0034] Für lange Datenvektoren, wie z. B. seismische Spuren, wird eine einzelne diskrete Kosinustransformation des gesamten Vektors wohl kaum viele vernachlässigbare Transformationskoeffizienten ergeben. Daher wird wie bei der JPEG ein blockierter DCT verwendet, wobei die Transformationslänge M = 8 beträgt und Fülldatenvektoren je nach Bedarf bis zu einer Länge N, die ein Mehrfaches von 8 ist, auf Null gesetzt werden. Die einer invertierten blockierten DCT-III entsprechende Matrix ist in Fig. 6 für N = 32 und M = 8 dargestellt. Außer, dass sie zur Kompression geeigneter ist, ist die Blocktransformation auch effizienter, und zwar mit einem Rechenaufwand, der lediglich linear mit der Länge N der Datenvektoren zunimmt.

[0035] Wie die frühere DCT-II-Tansformation, die in JPEG-Kompression verwendet wird, besitzt die DCT-III mehrere nützlich Eigenschaften. Zuerst transformiert eine DCT-III reele Zahlen in reele Zahlen, so dass keine komplexe Arithmetik notwendig ist. Auch ist die DCT-III wie die frühere DCT-II eine unitäre Transformation: $CII^{-1} = C_{III}^{T}$. [Diese Eigenschaft wird von den obigen Gleichungen (1) impliziert]. Die der Fig. 6 entsprechende Vorwärtsblockmatrix DCT-III ist einfach die transponierte der dort dargestellten invertierten DCT-III-Blockmatrix.

[0036] Einen weitere nützlich Eigenschaft ist, dass eine invertierte DCT-III einer DCT-II-Vorwärtsmatrix äquivalent ist: $C_{III}^{-1} = C_{II}$. Dieses Verhältnis zwischen der DCT-III und der DCT-II und eine Wahl der Transformationslänge von M = 8 ermöglicht die Verwendung der hoch optimierten DCT-II-Algorithmen, die in der JPEG-Kompression verwendet werden, indem einfach die Algorithmen für Vorwärts- und Inverstransformation vertauscht werden.

[0037] Wenn sämtliche oben genannten Eigenschaften nützlich sind, warum verwendet man dann nicht einfach die frühere DCT-II der JPEG? Die Antwort liegt in einer erfindungsgemäßen Abwandlung der DCT-III, um die von der JPEG-Kompression verursachten Blockartefakte zu vermeiden.

[0038] Blockartefakte treten auf, wenn lediglich ein Teilsatz der Spalten der in Fig. 6 gezeigten Matrix bei einer Appromixation eines Vektors verwendet wird. Man nehme z. B. an, dass solch ein Vektor hoch komprimiert ist, indem lediglich zwei Spalten mit den Indizes k = 11 und k = 19 verwendet werden. Wie in Fig. 7 dargestellt ergibt jede nicht triviale Kombination dieser beiden Spalten eine Unstetigkeit zwischen den Probenindizes j = 15 und j = 16 in der Approximation. Solche Unstetigkeiten erzeugen die Blockartefakte, die in Bildern sichtbar sind, welche unter Verwendung des JPEG-Algorithmus komprimiert wurden.

Gefaltete DCT-III

[0039] Um bei der Kompression die durch Verwendung einer Block-DCT-III verursachten Artefakte zu vermeiden, werden die in Fig. 7 gezeigten Blockkosinusse mit den in Fig. 8 dargestellten gleichmäßig zusammenlaufenden Kosinussen ersetzt. Man vergleiche die in Fig. 9 dargestellte vollständige invertierte Transformationsmatrix mit der invertierten DCT-III-Blockmatrix der Fig. 6. Die Anzahl der Proben in jedem Kosinus ist (außer an den Enden) auf 16 angestiegen, so dass Kosinusse in benachbarten Blöcken sich nun überlappen und dass jeder Kosinus gleichmäßig auf Null zusammenläuft. Die Gewichte der Summe dieser Kosinusse wird die in Fig. 7 gezeigten Unstetigkeit nicht erzeugen.

[0040] Die diesen zusammenlaufenden Kosinussen entsprechenden Transformationen werden häufig lokale Kosinustransformationen oder überlappte orthogonale Transformationen genannt (Wickerhauser, 1994, Seite 370; es besteht eine Ähnlichkeit zwischen Fig. 8 und Wickerhausers Fig. 11.5). Hier werden diese Transformationen als gefaltete Kosinustransformationen bezeichnet, um die Art und Weise wiederzuspiegeln, in welcher sie berechnet werden. Insbesondere ist die im erfindungsgemäßen Kompressionsalgorithmus verwendete Transformation eine gefaltete DCT-III-Transformation. Die inverse Transformation, die der in Fig. 9 gezeigten Matrix entspricht, wird eine invertierte gefaltet DCT-III-Transformation genannt.

[0041] Das Falten ist eine Art und Weise, um gleichmäßige, zulaufende 16-Proben-Kosinusse unter Verwendung hoch optimierter blockförmiger DCT-Algorithmen mit M = 8 zu erzielen. Wickerhauser (1994, Seite 103) beschreibt dieses Verfahren als eine „bemerkenswerte Beobachtung, die unabhängig von mehreren Personen gemacht wurde" und fährt damit fort, seine Anwendung auf die Kompression zu erörtern. Die bei der Kompression in dieser Erfindung verwendete Faltoperation ist eine von vielen von Wickerhauser beschriebenen, wurde jedoch durch die Arbeit von Jawerth und Sweldens (1995) und Jawerth et al. (1996) inspiriert. Letztere Autoren erörtern Aspekte des Faltens, die für die Kompression besonders relevant sind.

[0042] Die Geschicklichkeit und Effizienz des Faltens liegt in der Tatsache, dass die invertierte gefaltete DCT-III-Matrix, die in Fig. 9 gezeigt ist, das Produkt einer in Fig. 10 gezeigten Auffaltmatrix und der in Fig. 6 gezeigten invertierten DCT-III-Blockmatrix ist. Da jede Zeile und Spalte der Auffaltmatrix nicht mehr als zwei Elemente enthält, die nicht Null sind, ist der Rechenaufwand des Auffaltens eine fast unbedeutende Zugabe zum Rechenaufwand der invertierten DCT-III-Blockmatrix.

[0043] In der Praxis werden die in Fig. 6, Fig. 9 oder Fig. 10 dargestellten Matrizen nie wirklich konstruiert. Stattdessen werden Operationen auf Probenwerten durchgeführt, welche dieselbe Wirkung wie die Multiplikation mit diesen Matrizen besitzen. Für die invertierten DCT-III-Blockmatrix der Fig. 6 ist diese Operation der bei der JPEG-Kompression verwendete, hoch effiziente Vorwärtsalgorithmus der DCT-II-Blockmatrix. Für die Auffaltmatrix der Fig. 10 ist diese Operation ein paarweises Mischen der Probenwerte über die Blockgrenzen hinaus bei den Indizes j = 8, 16, ..., wie in Fig. 11 dargestellt.

[0044] Die Faltoperation ist einfach die Inverse der Auffaltoperation, ein anderes paarweises Mischen derselben Probenwerte. Da das Falten und Auffalten um die Blockgrenzen herum zentriert sind, werden diese Operationen am kompaktesten durch verschobene Probenwerte spezifiziert, welche definiert sind durch $y_l(j) = y(lM + j)$. Das Sympol l ist ein Blockindex und j ist eine Probe innerhalb eines Blockindex.

[0045] Dann wird das Falten erzielt durch

$$y_l(j) = f(j)x_l(j) + f(-j)x_l(-j)$$
$$y_l(-j) = f(j)x_l(-j) - f(-j)x_l(j)$$
$$l = 1, 2, ..., N/M - 1,$$

j = 1, 2, ..., M/2 – 1,
$y_l(j) = x_l(j)$, sonst (3a)

und das Auffalten wird erzielt durch

$x_l(j) = f(j)y_l(j) – f(–j)y_l(–j)$, $x_l(–j) = f(j)y_l(–j) + f(–j)y_l(j)$ l = 1, 2, ..., N/M – 1, j = 1, 2, ..., M/2 – 1, $x_l(j) = y_l(j)$, sonst (3b)

wobei die Faltfunktion f(j) definiert ist durch

$$f(j) = \sin\left[\frac{\pi}{4}\left(1+\frac{2j}{M}\right)\right] \qquad (3c)$$

[0046] Die gleichmäßig zulaufenden Kosinusse in Fig. 8 und die Matrix solcher Kosinusse in Fig. 9 wurden berechnet, indem die Gleichungen (3b) auf die Spalten der in Fig. 6 gezeigten Matrix angewendet wurden.

[0047] Die gefaltete DCT-III-Vorwärtsmatrix der Probenwerte x(j) wird bestimmt, indem zuerst y(j) über die Gleichungen (3a) berechnet wird und dann für jeden Block aus 8 Proben z(k) durch die Gleichung (1a) berechnet wird. Ebenso wird die invertierte gefaltete DCT-III-Matrix bestimmt, indem zuerst y(j) über Gleichungen (1b) für jeden Block berechnet wird und dann x(j) über Gleichungen (3b) berechnet wird.

[0048] Wie von Wickerhauser (1994, Seite 105), Jawerth und Sweldens (1995) und Jawerth et al. (1996) besprochen wird, sind viele alternative, jedoch ähnliche Faltoperationen möglich. Die von den Gleichungen (3) definierte Falteoperation wird aus zwei Gründen gewählt.

[0049] Zuerst ist die Faltoperation unitär. Die in Fig. 10 gezeigte Auffaltmatrix ist die transponierte der entsprechenden Faltmatrix (nicht gezeigt). Um diese Eigenschaft zu beweisen, beobachte man, dass die Faltfunktion der Gleichung (3c) die Bedingung $f^2(j) + f^2(–j) = 1$ erfüllt, und man drücke dann die Falt- und Auffaltoperationen der Gleichungen (3a) und (3b) als Multiplikationen mit 2×2 Matrizen aus. Man invertiere analytisch die 2×2 Faltmatrix, um zu sehen, dass ihre inverse gleich ihrer transponierten ist, was gleich der 2 × 2 Auffaltmatrix ist. Die gesamte Faltoperation ist daher unitär, da sie aus diesen 2×2 Mischungen von Probenwerten über Blockgrenzen hinaus besteht.

[0050] Zweitens stellt die erfindungsgemäße Faltoperation sicher, dass eine konstante Funktion, wie z. B. x(j) = 1 Transformationskoeffizienten $z_l(k)$ in jedem Block ergibt, die lediglich für k = 0 nicht Null sind, wodurch die Kompression konstanter (oder langsam variierender) Daten verbessert wird. In der Terminologie von Jawert et al. (1996) erzielt die gefaltete DCT-III-Vorwärtsmatrix die „Auflösung der Konstanten".

[0051] Um diese zweite Eigenschaft der gefalteten DCT-III-Matrix zu beweisen, wende man analytisch die Faltoperation der Gleichungen (3a) auf konstante Probenwerte xl(j) = 1 an und verifizieren, dass das Ergebnis

$$y_1(j) = \sqrt{MC_{III}(0j)}$$

ist, wobei $C_{III}(kj)$ in Gleichung (2) definiert ist. In anderen Worten wird die Faltfunktion so gewählt, dass konstante Probenwerte so gefaltet werden, dass sie (bis auf einen Skalierungsfaktor von $\sqrt{M}$) dem ersten Kosinus (k = 0) der DCT-III-Matrix entsprechen. Da dieser Kosinus senkrecht zu allen anderen Kosinussen jener Transformation ist, werden lediglich die k = 0 Transformationskoeffizienten in jedem Block nach der gefalteten DCT-III-Vorwärtsmatrix nicht Null sein. Der Wert jedes von Null verschieden Koeffizienten wird $zl(0) = \sqrt{M}$ sein.

[0052] Genauer gesagt gilt diese Eigenschaft für alle Blöcke, außer dem ersten und dem letzten. Obwohl die Auflösung der Konstanten für den ersten und den letzten Block erhalten werden kann, indem die Falt- und Auffaltmatrizen so modifiziert werden, dass sie in ihren Ecken nicht unitär sind, wird eine geringere Kompression in diesen Blöcke akzeptiert und eine strikt unitäre Falt- und Auffaltoperation wird beibehalten.

Vergleich mit dem in der JPEG-Kompression verwendeten DCT-II-Algorithmus

[0053] Die DCT-II-Vorwärtsmatrix, die in der JPEG-Kompression verwendet wird, erzielt die Auflösung der Konstanten ohne Falten, da der erste Kosinus der PCT-II (z. B. die erste Reihe der Matrix in Fig. 5) konstant ist. Tatsächlich ist diese Eigenschaft der Grund, warum die DCT-II und nicht die DCT-III im JPEG-Kompressionsstandard spezifiziert ist. Die Verwendung des Falttricks mit der DCT-II könnte in Betracht gezogen werden,

da das Falten vor der JPEG-Kompression und das Auffalten nach der JPEG-Dekompression durchgeführt werden könnte, ohne den Standard-JPEG-Algorithmus abzuwandeln.

[0054] Leider sind die Falt- und Auffaltoperationen der Gleichungen (3) nicht für die DCT-II geeignet. Insbesondere würde die Gleichungen (3b) keine gleichmäßig zusammenlaufenden Kosinusse ergeben, wie jene in Fig. 8 dargestellt.

[0055] Jawerth et al. (1995) beschreiben alternative Falt- und Auffaltoperationen, die für die DCT-II geeignet sind, und die die Auflösung der Konstanten beibehalten. Jedoch sind diese Operationen nicht unitär. Sie sind in der Tat schlecht konditioniert und neigen dazu, die Unstetigkeiten in den Probenwerten, die in der Nähe der Blockgrenzen auftreten, zu verstärken und dadurch die Effizienz der Kompression zu verringern.

[0056] Im Gegensatz dazu sind die Falt- und Auffaltoperationen der Gleichungen (3) unitär, ebenso wie die vorwärts gerichteten und invertierten DCT-III-Matrizen der Gleichungen (1). Wenn F die Faltmatrix bezeichnet, dann kann die gefaltete DCT-III-Transformation ausgedrückt werden als $z = C_{III}Fx$, und man beobachte, dass

$$\begin{aligned} z^T z &= x^T F^T C_{III}{}^T C_{III} F x \\ &= x^T F^{-1} C_{III}{}^{-1} C_{III} F x \\ &= x^T x \end{aligned}$$

$$\sum_{k=0}^{N-1} z^2(k) = \sum_{j=0}^{N-1} x^2(j) \qquad (4)$$

[0057] In anderen Worten ist die Summe der hoch zwei genommen Probenwerte nach der gefalteten DCT-III-Transformation gleich jener Summe vor der gefalteten DCT-III-Transformation. Diese Eigenschaft kann verwendet werden, um die Verzerrung in den dekomprimierten Daten zu schätzen, die durch die Quantisierung der Transformationskoeffizienten verursacht wird.

Quantisierung und Kodierung

[0058] Wie oben beschrieben, ist die im Kompressionsalgorithmus der Erfindung verwendete Transformation einfach die Inverse jener, die bei der JPEG-Kompression verwendet wird, wobei jedoch das Falten eingeschlossen wird, um Blockartefakte zu verringern. Die verwendeten Quantisierungs- und Kodierverfahren wurden auch von jenen bei der JPEG-Kompression verwendeten adaptiert. Die Unterschiede zwischen den JPEG-Verfahren und der Erfindung werden im Folgenden beschrieben.

Quantisierung

[0059] Da die JPEG-Kompression für Bilder gedacht ist, sind die Daten vor der Kompression entweder 8 Bit- oder 12 Bit-wertig (Farbbilder werden durch rot-grün-blau-Triplets solcher Proben dargestellt). Nach einer 2D-Transformation mit Hilfe der DCT-II können 8 Bit Daten bis zu 11 Bits pro Probe erfordern, da der größte Probenwert nach einer 2D-DCT-II bis zu 8 mal größer als jener vor der Transformation ist. Im Allgemeinen beträgt der größte Probenwert nach der DCT-II bis zu $M^{D/2}$ mal den Größenwert vor der Transformation, wobei D die Anzahl der transformierten Dimensionen ist. In anderen Worten gilt

$$|z|_{max} \leq M^{D/2}|x|_{max} \qquad (5)$$

[0060] Diese obere Grenze wird erricht, wenn die Daten exakt einem der in der DCT-II verwendeten Kosinussen entsprechen, wie z. B. dem konstanten ersten Kosinus $C_{II}(k = 0j)$.

[0061] Derselbe Faktor $M^{D/2}$ gilt auch für die gefaltete DCT-III-Transformation, die im Kompressionsalgorithmus dieser Erfindung verwendet wird. Nach dem Transformieren von ganzzahligen 8 Bit Daten mit einer gefalteten DCT-III-Transformation quantisiert und kodiert daher das Verfahren der Erfindung 11 Bit Zahlen in 2D-Kompression und 13 Bit-Zahlen in 3D-Kompression.

[0062] Der Quantisierungsschritt sowohl im JPEG-Kompressionsalgorithmus als auch im Verfahren der Erfindung ist eine Skalierungsoperation, die dafür konzipiert ist, die Anzahl der zu kodierenden Bits zu verringern, und es ist diese Verringerung der Anzahl der Bits, die für die Verzerrung bei Daten verantwortlich ist, die komprimiert und dann dekomprimiert werden. Bei solchen Verlust behafteten Kompressionsverfahren wird diese

Verzerrung als Gegenleistung für höhere Kompressionsverhältnisse akzeptiert.

[0063] Bei der JPEG-Kompression werden die Transformationskoeffizienten in jedem Block anders quantisiert, wobei hohe Wellenzahlen (räumlich Frequenzen) mit weniger Bits als niedrige Wellenzahlen dargestellt werden. Die Wellenzahl abhängige Skalierung bei der JPEG-Kompression wird gewöhnlich für die menschliche Sichtwahrnehmung optimiert.

[0064] Bei dem Kompressionsverfahren der Erfindung werden alle Wellenzahlen gleich quantisiert. Es werden keine Wellenzahl abhängigen Fehler eingeführt. Ein Grund dafür ist, dass bei seismischen Daten oft hohe Wellenzahlen von großem Interesse sind. Z. B. entsprechen mit seismischen Daten abgebildete Verwerfungen hohen Wellenzahlen. Ein anderer Grund ist, dass seismischen Daten häufig von Computeralgorithmen analysiert werden, die vom menschlichen Sehsystem unabhängig sind.

Lokale Quantisierung

[0065] Beim Komprimieren von 8 Bit Bilddaten liegen die Probenwerte zwischen –128 und +127; es kann daher angenommen werden, dass Datenblöcke geringer Amplitude nicht signifikant sind und während der Kompression getrost auf Null quantisiert werden können. Insbesondere der JPEG-Standard macht diese Annahme, da er lediglich einen Satz von Quantisierungsskalierungsfaktoren für ein gesamtes Bild erlaubt. Wie oben besprochen, können diese Skalierungsfaktoren für verschiedene Koeffizienten (verschiedene Wellenzahlen) in einem Block variieren, jedoch wird derselbe Satz für jeden Block verwendet. In diesem Sinne ist die JPEG-Quantisierung global.

[0066] Beim Komprimieren von 32 Bit-Gleitkommadaten wird bevorzugt, eine lokale Quantisierung durchzuführen, und zwar mit Skalierungsfaktoren, die von Block zu Block variieren. Vor der Kompression solcher Daten ist es möglich, dass die maximale Probenamplitude $|x|_{max}$ unbekannt ist, und das Auslesen jeder Probe vor der Kompression zur Bestimmung dieses Wert kann kostspielig sein. Des Weiteren kann streng nicht angenommen werden, dass niedrige Amplituden insignifikant sind; insbesondere seismische Daten erfordern häufig eine beträchtliche Verarbeitung bevor diese Annahme gültig ist. Obwohl daher ein einziger Skalierungsfaktor verwendet wird, um alle Transformationskoeffizienten in einem Block zu quantisieren, kann erlaubt werden, dass ein Skalierungsfaktor von Block zu Block variiert. Insbesondere bezeichnet in der obigen Gleichung (5) $|x|_{max}$ den Maximalkoeffizienten in jedem transformierten Block und ein unterschiedlicher Skalierungsfaktor s wird für jeden Block berechnet.

[0067] Die lokale Quantisierung ergibt geringere Kompressionsverhältnisse (erzeugt mehr Bits pro Probe) als die globale Quantisierung. Ein offensichtlicher Grund ist, dass zusätzlich Bit erforderlich sind, um die Quantisierungsskalierungsfaktoren für jeden komprimierten Block zu speichern. Ein weniger offensichtlicher Grund ist, dass die lokale Quantisierung weniger Proben auf Null quantisieren kann, als die globale Quantisierung. Daher ist eine Auswahl von entweder der lokalen oder der globalen Quantisierung im Kompressionsalgorithmus der Erfindung erlaubt.

[0068] Die lokale Quantisierung verarbeitet in einem einzelnen Block keinen großen dynamischen Bereich. Z. B. können in unverarbeiteten seismischen Daten Reflexionen mit niedriger Amplitude unter Oberflächenwellen mit hoher Amplitude versteckt sein. In mit Rauschen von hoher Amplitude kontaminierten Blöcken kann der Kompressionsalgorithmus der Erfindung selbst mit lokaler Quantisierung ein Signal mit geringer Amplitude auf Null quantisieren, so dass das Signal durch eine Verarbeitung nach der Dekompression nicht mehr hinterhergestellt werden kann. Daher sollte ein Rauschen hoher Amplitude vor der Kompression abgeschwächt werden.

Komprimierter virtueller Speicher

[0069] Ein virtueller Speicher gibt die Illusion eines Speichers, der größer als der physikalisch verfügbare ist. Diese Illusion ist für jene Anwendungen am wirkungsvollsten, die dazu neigen, auf Daten zuzugreifen, die in der Nähe anderer Daten liegen, auf welche kürzlich zugegriffen wurde. Solche Anwendungen besitzen eine gute Referenzlokalität.

[0070] Anwendungen, die mit 2D- oder 3D-Arrays arbeiten, weisen häufig eine gute Referenzlokalität auf. Z. B. kann eine Anwendung zur seismischen Interpretation aufeinanderfolgende 2D-Datenschnitte auf einem 3D-Array anzeigen. Mit einem ausreichend schnellen lokalen Dekompressionsalgorithmus können auf Wunsch die Blöcke dekomprimiert werden, die die Proben für solche Schnitte enthalten, ohne das gesamte 3D-Array zu dekomprimieren. Der Kompressionsalgorithmus der Erfindung ist insbesondere bei solchen Anwendungen

von Nutzen.

[0071] Der Schlüssel zu solchen Anwendungen ist die Fähigkeit, einen Teilsatz eines großen Arrays zu komprimieren oder zu dekomprimieren, ohne das gesamte Array zu komprimieren oder zu dekomprimieren. Für die auf einer DCT-II-Blocktransformation basierende Kompression, wie jene, die in der JPEG-Kompression verwendet wird, ergibt sich dieses Merkmal von selbst. Um eine einzige Probe zu dekomprimieren, ist es lediglich notwendig, den Block zu dekodieren, dequantisieren und invers DCT-II zu transformieren, welcher die Probe enthält. Sobald die Dekompression für eine Probe erfolgt ist, ist die Dekompression aller Proben ihres Blocks erreicht. Unter Annahme der Referenzlokalität wird der Rechenaufwand der Dekompression der anderen Proben in jenem Block nicht vergeudet.

[0072] Für den Kompressionsalgorithmus der Erfindung, der auf einer gefalteten DCT-III-Transformation basiert, ist etwas zusätzliche Arbeit notwendig. Man nehme die 2D-Kompression und die vier Blöcke von 8×8 Proben, die in Fig. 12 dargestellt sind. Um die Probe zu dekomprimieren, die dem gefüllten Kreis in Block A entspricht, ist es notwendig (1) alle vier Blöcke (A, B, C, D) zu dekodieren, zu dequantisieren und invers zu DCT-III transformieren, und (2) die den gefüllten Kreisen entsprechenden vier Proben aufzufalten. Das Auffalten über die Blockgrenzen ist genau wie in Fig. 11 dargestellt und von Gleichung (3b) beschriebene, und wird zuerst für eine Dimension und dann für die andere durchgeführt. Obwohl vier Blöcke notwendig sind, ist die meiste Arbeit, die notwendig ist, um benachbarte Proben zu dekomprimieren, bereits durchgeführt, sobald die Dekompression der Probe in Block A erreicht ist. Wiederum unter Annahme der Referenzlokalität wird diese zusätzliche Rechenleistung nicht vergeudet.

[0073] Ein weiterer Unterschied zwischen dem Quantisierungsschritt der Erfindung und jenem der JPEG rührt aus der Notwendigkeit her, 32 Bit Gleitkommadaten zu quantisieren, die einen viel größeren dynamischen Bereich als 8- oder 12-Bit Bilddaten besitzen. Um einen Gleitkommawert z in eine ganze Zahl i mit B + 1 Bit (inklusive einem Vorzeichenbit) zu quantisieren, wird der folgende Algorithmus verwendet

$$i = \begin{cases} \lfloor z \times s + 1/2 \rfloor, & z \geq 0, \\ \lceil z \times s - 1/2 \rceil, & z > 0, \end{cases} \qquad (6)$$

wobei s der Quantisierungsskalierungsfaktor ist. Um eine Überlauf zu vermeiden, ist $|i| < 2^B$ erforderlich. Diese Einschränkung und die Gleichung (6) führen zur folgenden Gleichung für den Skalierungsfaktor:

$$s = \frac{\left(2^B - 1/2\right)(1-\varepsilon)}{|z|_{\max}} \qquad (7)$$

wobei ε ein Gleitkommaepsilon ist, d. h, die kleinste positive Zahl, die von 1 unter Verwendung von Gleitkommaarithmetik abgezogen werden kann, um eine Zahl ungleich 1 zu erhalten.

Huffman-Kodierung

[0074] Der JPEG-Kompressionsstandard erlaubt zwei Verfahren zum Kodieren der durch die Quantisierung erzeugten ganzen Zahlen – Huffman-Koderung und arithmetische Kodierung – und viele andere Kodierverfahren sind möglich. Die JPEG-Huffman-Kodierung wird im Kompressionsalgorithmus der Erfindung verwendet, da sie von der Berechnung her schnell, einfach umzusetzen und frei verfügbar ist.

[0075] Mit einer Ausnahme folgen die Huffman-Kodierungs- und Dekodierungsalgorithmen der JPEG-Spezifikation. Der JPEG-Standard spezifiziert eine spezielle Kodierung der DC-Transformationskoeffizienten (k=0). Die JPEG-Kompression nutzt die Tatsache aus, dass diese DC-Koeffizienten häufig stark mit benachbarten Blöcken korrelieren. Diese spezielle Behandlung wird im Verfahren der Erfindung aus zwei Gründen nicht verwendet: (1) sie führt Abhängigkeiten von Block zu Block ein, die die Kompression und Dekompression der einzelnen Blöcke komplizierter machen, und (2) seismische Daten besitzen typischerweise relativ kleine DC-Koeffizienten (die DC-Koeffizienten sind, obwohl klein, aufgrund der verwendeten kurzen Transformation mit M = 8 selten vernachlässigbar). Daher wird der DC-Koeffizient genauso wie die anderen (AC-)Koeffizienten kodiert.

Lokale Kompression

[0076] Der Hauptvorteil eins JPEG-ähnlichen Algorithmus ist, dass ein Teil des mehr-dimensionalen Arrays komprimiert werden oder verwendet werden kann, ohne es in seiner Gesamtheit zu verarbeiten. Im Gegensatz dazu fehlt dieses Merkmal den auf Wavelet-Transformationen basierenden Kompressionsalgorithmen (z. B.

Bradley et al., 1993; Wickerhauser, 1994). Während der JPEG-Standard dieses Merkmal nicht explizit unterstützt, macht es die in der JPEG-Kompression verwendete DCT-II-Blocktransformation möglich. Diese Fähigkeit wird im Kompressionsalgorithmus der Erfindung, die auf einer gefalteten DCT-III basiert, ausgenutzt.

[0077] Jeder Datenblock in einem Array kann als einer Seite des virtuellen Speichers analog angesehen werden. Für die 2D-Kompression würde jede Seite 64 = 8 × 8 Proben enthalten; für die 3D-Kompression würde jede Seite 512 = 8 × 8 × 8 Proben enthalten. Proben werden dekomprimiert, wenn sie eingestellt werden, und Wellenzahl-komprimiert, wenn sie herausgenommen werden. Ein Arbeitssatz unkomprimierter Seiten wird im Speicher gehalten, während die meisten Seiten komprimiert bleiben und entweder im Speicher oder auf Platte gespeichert sind (wenn sie auf Platte gespeichert sind, können Seiten kombiniert werden, um die Ein-/Ausgabeeffizienz zu erhöhen). Wenn der Arbeitssatz groß genug ist, und wenn unsere Anwendung eine gute Referenzlokalität besitzt, wird der Rechenaufwand des Komprimierens und/oder Dekomprimierens jeder Seite über den Zugang zu den meisten Proben in jenen Seiten amortisiert.

Fazit

[0078] Die Adaptation des JPEG-Kompressionsalgorithmus gemäß der Erfindung macht es möglich, dass ein Großteil jenes Algorithmus wiederverwendet werden kann. Die JPEG-Verfahren werden für DCTs mit Länge 8 wiederverwendet, indem die Vorwärts- und inversen Transformationen einfach vertauscht werden. Das JPEG-Quantisierungsverfahren wird abgewandelt, um eine bevorzugte Behandlung kleiner Wellenzahlen zu vermeiden und um Block-zu-Block-Variationen in den Datenamplituden zu bearbeiten. Probenwerte werden vor einer Vorwärts-DCT auch über Blockgrenzen hinaus gefaltet und solche Werte werden nach einer inversen DCT aufgefaltet. Dieses Falten und Auffalten unterdrückt die Blockartefakte, die in Bildern sichtbar sind, welche mit dem JPEG-Algorithmus komprimiert wurden.

[0079] Zum Vergleich der Leistungsfähigkeit des Verfahrens der Erfindung mit anderen Algorithmen sind die Rechenzeit und die Verzerrung für ein spezifiziertes Kompressionsverhältnis zwei nützliche Maßstäbe. Vorläufige Benchmarks mit einer frühen Umsetzung des Algorithmus der Erfindung sind ermutigend. Die Rechenzeiten betragen ungefähr die Hälfte und die Verzerrungen sind fast identisch jenen für einen Waveletbasierten Algorithmus, und zwar für einen weiten Bereich von Kompressionsverhältnissen (Diller 1997, persönliche Mitteilung). Es wird angenommen, dass die Rechenzeiten für das Verfahren der Erfindung geringer als für Wavelet-basierte Verfahren sind (und zwar aufgrund von weniger Multiplikationen und Additionen und einem lokaleren Speichereinsatz).

Patentansprüche

1. Verfahren zum Komprimieren eines eindimensionalen Arrays x von N die Eigenschaften der Erde darstellenden Mustern, umfassend die folgenden Schritte:
(a) Unterteilen des Arrays x in Blöcke von M Mustern, wobei M < N ist, und gekennzeichnet durch
(b) Falten der Muster $x_l(j) = x(lM + j)$ über jede Blockgrenze 1, entsprechend

$y_l(j) = f(j)x_1(j) + f(-j)x_l(-j)$,
$y_l(-j) = f(-j)x_l(-j) - f(-j)x_l(j)$,
$1 = 1, 2, ..., N/M - 1$,
$j = 1, 2, ..., M/2 - 1$,
$y_l(j) = x_l(j)$ sonst,

wobei

$$f(j) = \sin\left[\frac{\pi}{4}\left(1+\frac{2j}{M}\right)\right],$$

(c) Transformieren der gefalteten Muster in jedem Block des Arrays y nach

$$z(k) = \sum_{j=0}^{M-1} y(j)b(j)\sqrt{\frac{2}{M}}\cos\left[\frac{\pi(2k+1)j}{2M}\right],$$

k = 0, 1, ..., M-1

wobei

$$b(j) = \frac{1}{\sqrt{2}}, j = 0$$

b (j) = 1 sonst
(d) Digitalisieren der transformierten Muster in jedem Block des Arrays z, um ganze Zahlen zu erzeugen, und
(e) Kodieren der ganzen Zahlen in einen Strom aus das komprimierte Array darstellenden Bits.

2. Verfahren nach Anspruch 1, angewendet auf ein mehrdimensionales Array, wobei der Blockschritt (a) der Faltschritt (b) und der Transformationsschritt (c) als Kaskade von Operationen entlang jeder Array-Dimension angewendet werden.

3. Verfahren nach Anspruch 2, wobei eine Untermenge des mehrdimensionalen Arrays komprimiert wird.

4. Verfahren nach Anspruch 3, wobei das mehrdimensionale Array seismische Signale darstellt.

5. Verfahren nach Anspruch 1, weiter umfassend das Dekomprimieren des komprimierten Arrays durch Umkehren der Schritte (a) bis (e) in umgekehrter Reihenfolge.

6. Verfahren nach Anspruch 5, angewendet auf ein mehrdimensionales Array, wobei die Umkehrung des Blockschritts (a), des Faltschritts (b) und des Transformationsschritts (c) als Kaskade von Operationen entlang jeder Arraydimension angewendet werden.

7. Verfahren nach Anspruch 6, wobei eine Untermenge des mehrdimensionalen Arrays dekomprimiert wird.

8. Verfahren nach Anspruch 7, wobei das mehrdimensionale Array seismische Signale darstellt.

9. Computerprogramm, das einen Satz von Computeranweisungen zum Komprimieren eines eindimensionalen Datenarrays x von N die Eigenschaften der Erde darstellenden Mustern kodiert, welches beim Ausführen auf einem Computer geeignet ist, das Datenkompressionsverfahren nach einem der Ansprüche 1 bis 8 durchzuführen.

Es folgen 6 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1

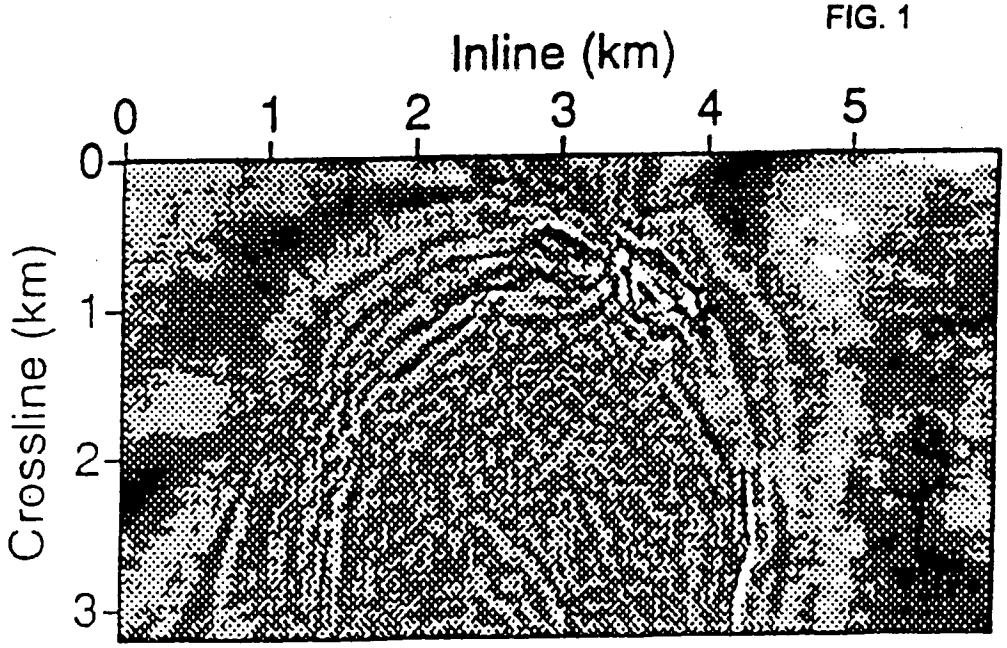
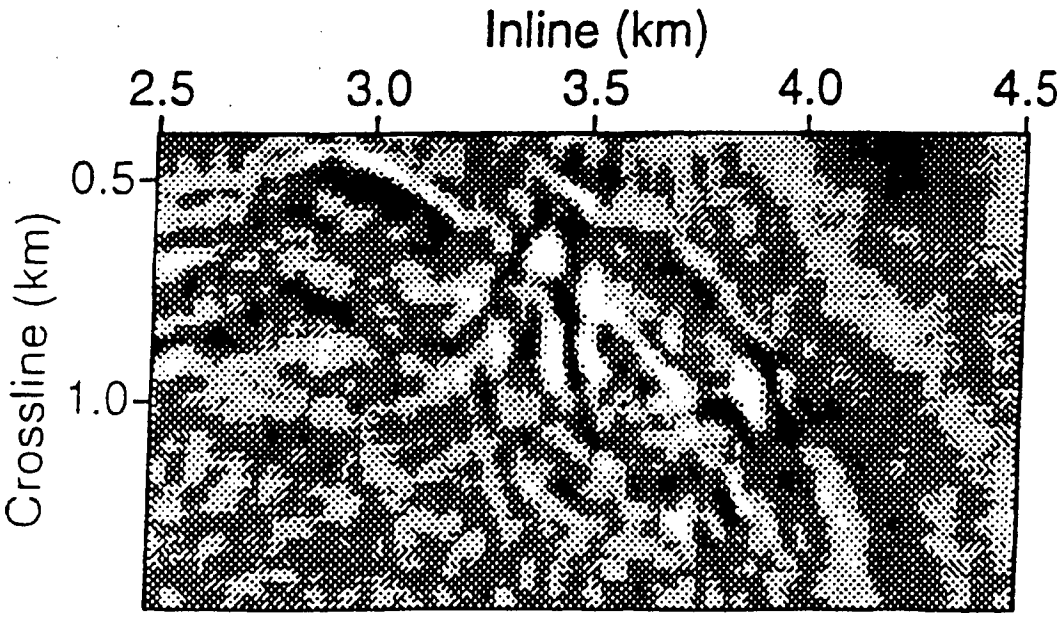


FIG. 2



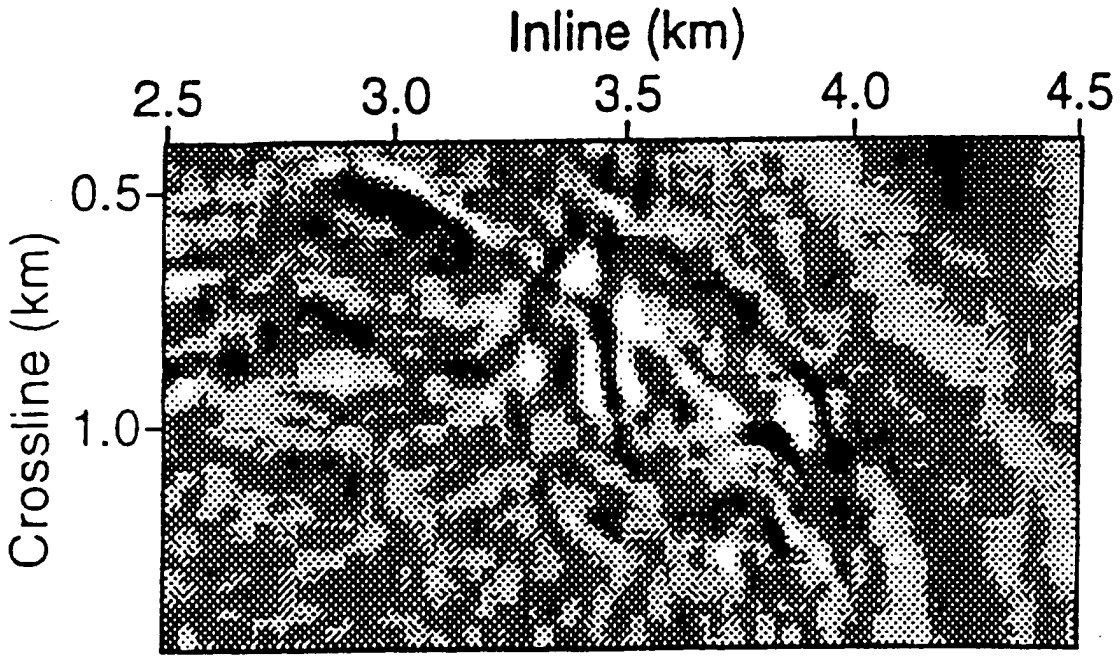

Inline (km)
2.5
3.0
3.5
4.0
4.5
0.5
1.0
Crossline (km)

FIG. 3.

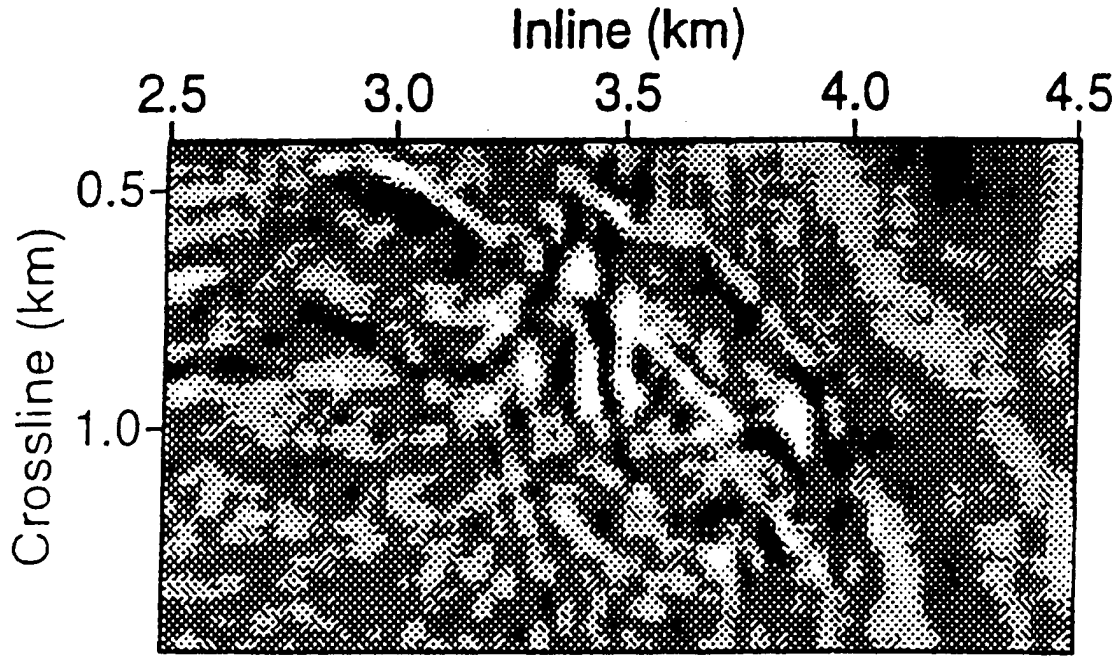

Inline (km)
2.5
3.0
3.5
4.0
4.5
0.5
1.0
Crossline (km)

FIG. 4.

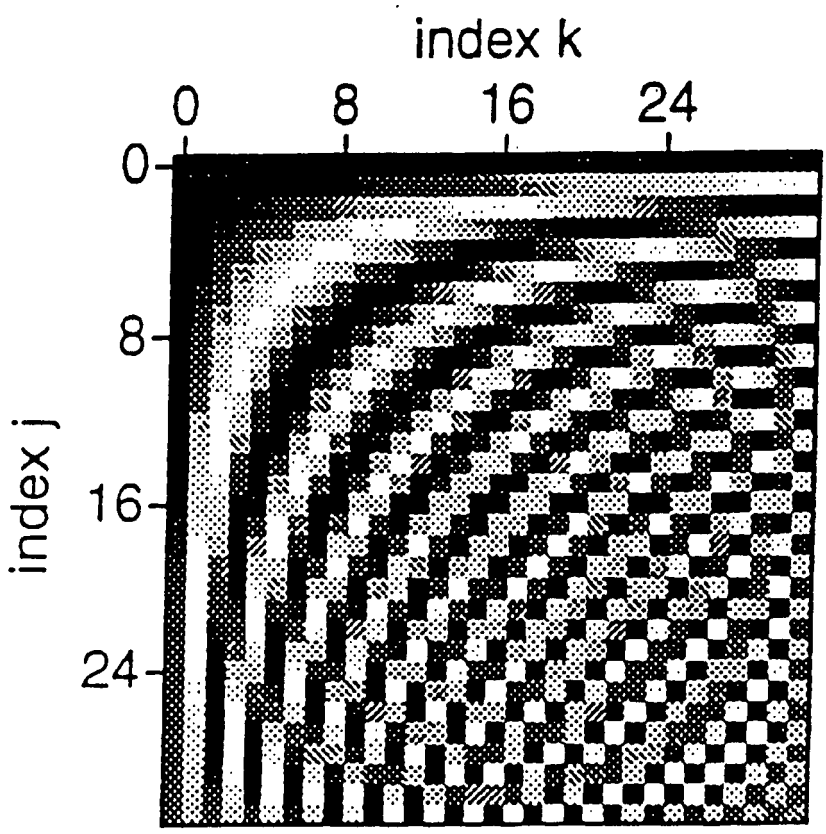


FIG. 5.

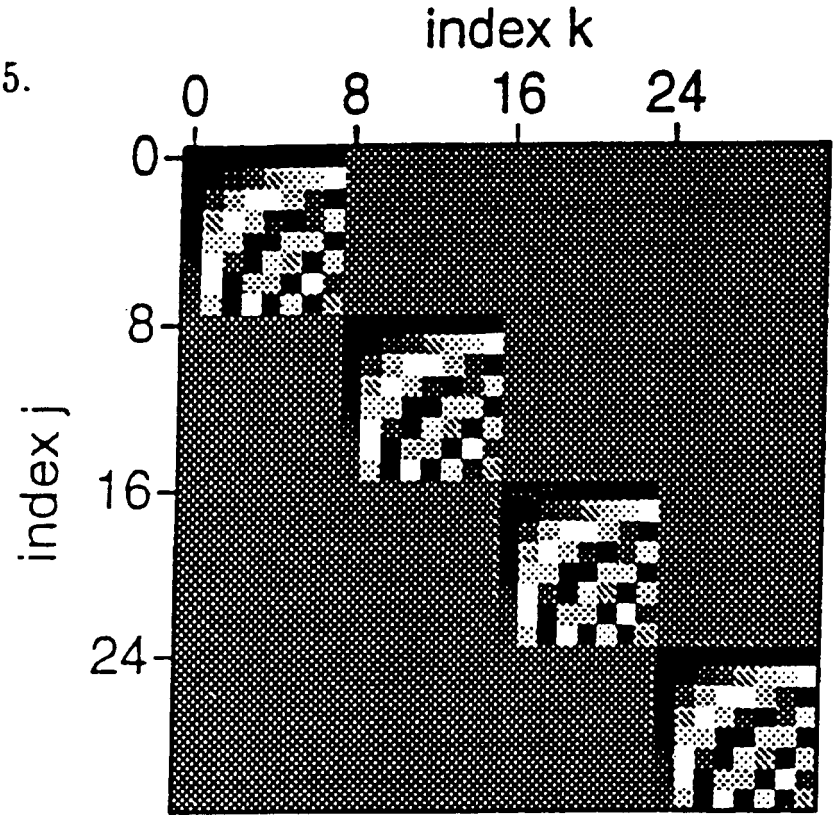


FIG. 6.

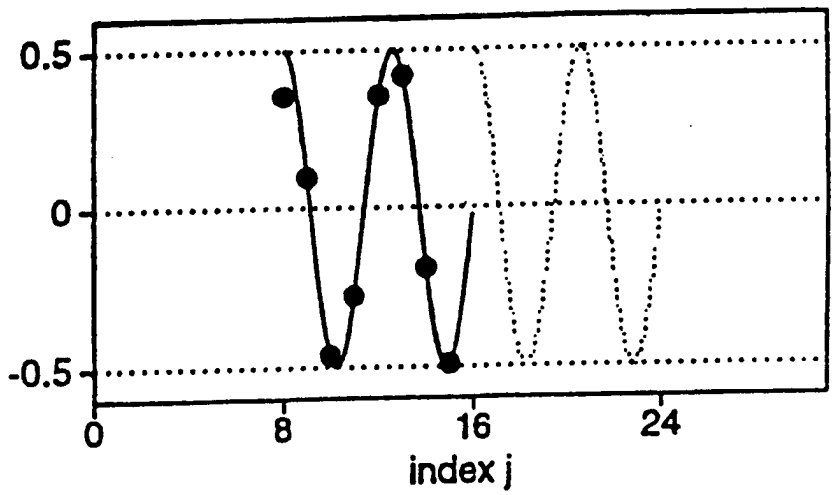


FIG. 7.

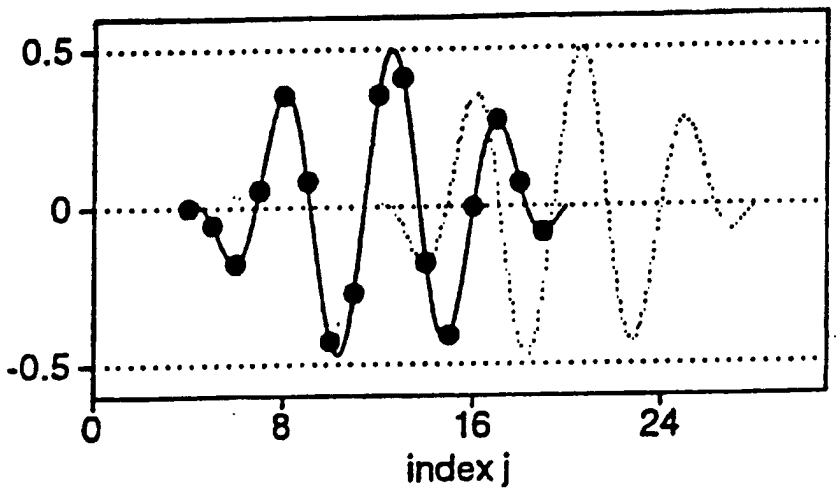


FIG. 8.

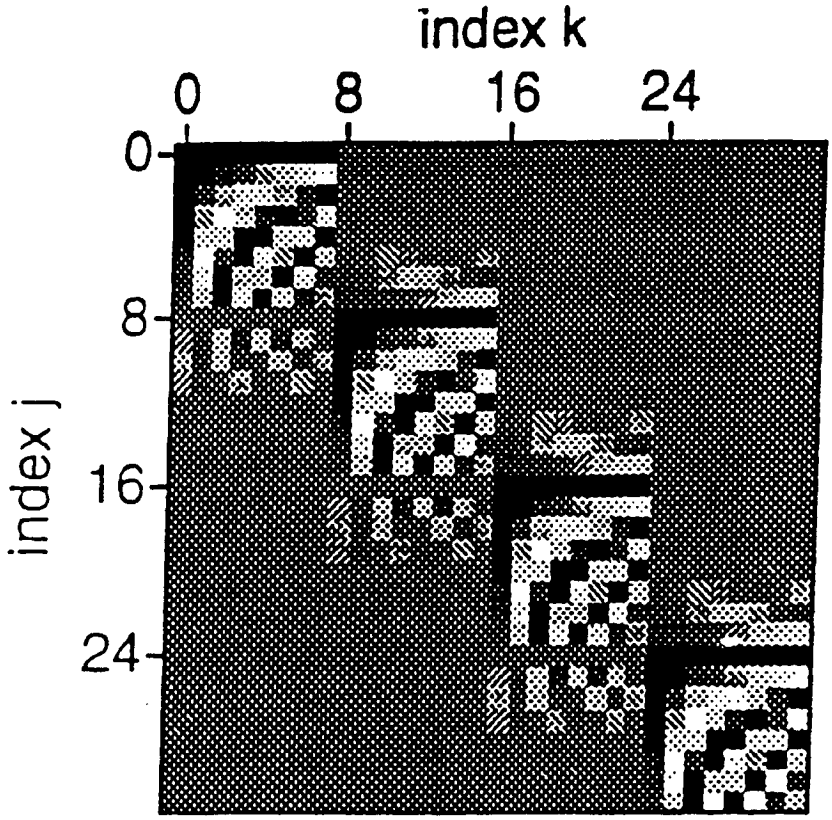


FIG. 9.

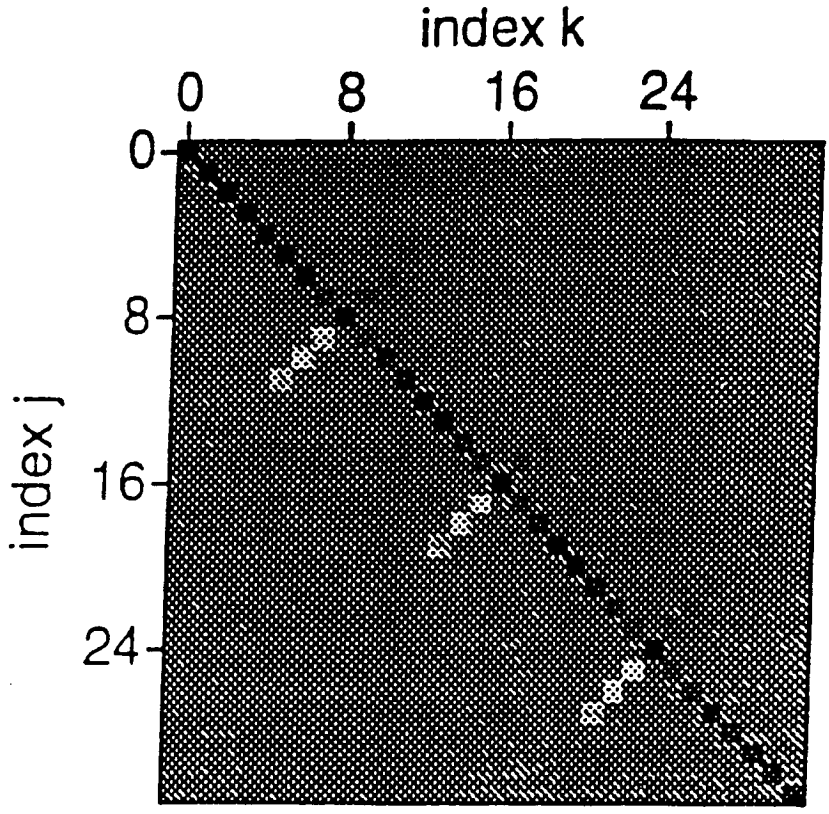


FIG. 10.

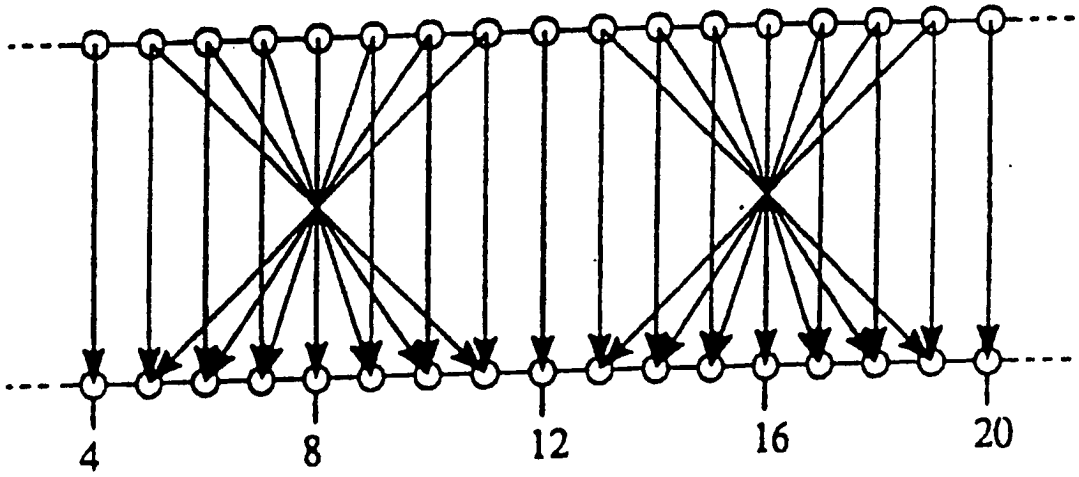


FIG. 11.

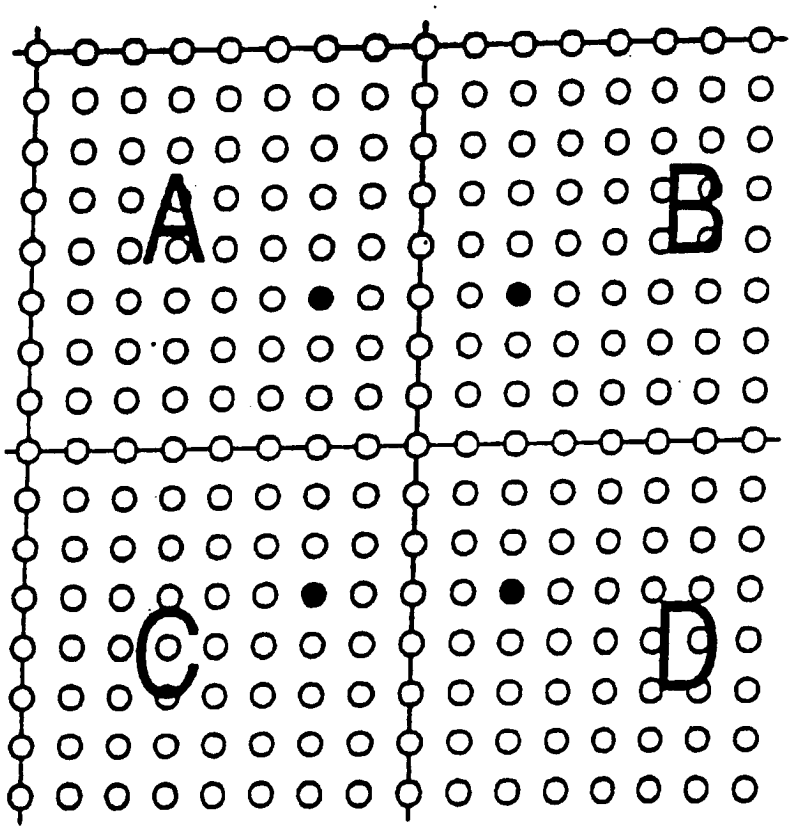


FIG. 12.