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

(10) **DE 603 08 255 T2** 2006.12.28

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 359 546 B1**
(21) Deutsches Aktenzeichen: **603 08 255.6**
(96) Europäisches Aktenzeichen: **03 006 487.7**
(96) Europäischer Anmeldetag: **21.03.2003**
(97) Erstveröffentlichung durch das EPA: **05.11.2003**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **13.09.2006**
(47) Veröffentlichungstag im Patentblatt: **28.12.2006**

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

(30) Unionspriorität:

377298 P	**02.05.2002**	**US**
376147 P	**28.02.2003**	**US**

(84) Benannte Vertragsstaaten:
AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LI, LU, MC, NL, PT, SE, SI, SK, TR

(73) Patentinhaber:
Microsoft Corp., Redmond, Wash., US

(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80538 München

(72) Erfinder:
Sridhar Srinivasan, Washington 98109, US;
Shankar Regunathan, Washington 98007, US

(54) Bezeichnung: **2D-Transformationen zur Bild- und Videokodierung**

Beschreibung

[0001] Die vorliegende Erfindung betrifft Verfahren zum digitalen Kodieren und Verarbeiten von Signalen. Im Besondern betrifft die Erfindung die Erstellung und die Verwendung einer Klasse von recheneffizienten Transformationen beim Kodieren und Dekodieren von Signalen wie beispielsweise die von Bild und Video.

[0002] Die Transformationskodierung ist ein Kompressionsverfahren, das in vielen Audio-, Bild- und Video-Kompressionssystemen angewendet wird. Nicht-komprmierte digitale Bild- und Videodaten werden typischerweise als Samples von Bildelementen oder Farben an Orten in einem Bild- oder Videoframe repräsentiert oder erfasst, der in einem zweidimensionalen Gitter angeordnet ist. So besteht beispielsweise ein typisches Format für Bilder aus einem Stream von 14-Bit-Farbbildelement-Samples, die als ein Gitter angeordnet sind. Jedes Sample ist eine Nummer, die die Farbenkomponenten an einem Pixelort in dem Gitter innerhalb eines Farbraums repräsentiert, wie beispielsweise RGB [Rot-Grün-Blau] oder der fast identische Farbraum YIQ. Verschiedene Bild und Videosysteme können unterschiedliche Farb-, Raum- und Zeitauflösungen für das Sampling verwenden.

[0003] Nicht-komprimierte digitale Bild- und Videosignale können eine enorme Speicher- und Übertragungskapazität verbrauchen. Durch Transformationskodierung wird die Größe der digitalen Bilder und Videos durch Transformieren der räumlichen Bereichsrepräsentation des Signals in eine Frequenzbereich- (oder andere ähnliche Transformationsbereiche) Repräsentation, und durch das anschließende Reduzieren der Auflösung von bestimmten im Allgemeinen weniger wahrnehmbaren Frequenzkomponenten der Transformationsbereichs-Repräsentation reduziert. Dadurch wird im Allgemeinen eine viel weniger wahrnehmbare Verschlechterung des digitalen Signals verglichen mit dem Reduzieren der Farb- oder Raumauflösung der Bilder oder Videos in dem räumlichen Bereich erzeugt.

[0004] Genauer gesagt, werden bei einem typischen Transformationskodierungsverfahren die nicht-komprimierten digitalen Bildpixel in zweidimensionale Blöcke einer festgelegten Größe unterteilt, wobei jeder Block möglicherweise mit anderen Blöcken überlappend angeordnet ist. Es wird eine lineare Transformation, bei der räumliche Frequenzanalyse durchgeführt wird, an einem jedem Block angewendet, wodurch die beabstandeten Samples innerhalb des Blockes in einen Satz von Frequenz- (oder Transformations-) Koeffizienten, die allgemeinen die Stärke des digitalen Signals in den entsprechenden Frequenzbändern über das Blockintervall hinweg darstellen, umgewandelt werden. Für die Kompression können die Transformationskoeffizienten selektiv Quantisierung unterzogen werden (das heißt, in ihrer Auflösung reduziert werden, wie bei dem Ablegen der am wenigsten signifikanten Bits der Koeffizientenwerte oder anderenfalls dem Abbilden der Werte in einer höheren Auflösungsanzahl, die auf eine geringere Auflösung eingestellt ist), uns sie können auch durch Entropiekodierung oder variable Längenkodierung in einen komprimierten Datenstream umgewandelt werden. Beim Dekodieren werden die Transformationskoeffizienten invers transformiert, um nahezu das ursprüngliche Farb-/Raum-gesamplete Bild-/Videosignal zu erzeugen.

[0005] Viele Bild- und Videokompressionssysteme wie beispielsweise MPEG und Windows Media wenden Transformationen auf Basis der Diskreten Kosinustransformation (DCT [Discrete Cosinus Transform]) an. Das DCT-Verfahren ist dafür bekannt, dass es günstige Energieverdichtungseigenschaften besitzt, die in einer fast optimalen Datenkompression resultieren. In diesen Kompressionssystemen wird die inverse DCT (IDCT) in den Rekonstruierungsschleifen sowohl in der Kodiereinrichtung als auch in der Dekodiereinrichtung des Kompressionssystems zum Rekonstruieren der einzelnen Bildblöcke angewendet. Eine exemplarische Implementierung der IDCT ist in „ISSS Standard Specification for the Implementations of 8 × 8 Inverse Discrete Cosine Transform.", IEEE Std. 1180-1990, 6. Dezember, 1990, beschrieben.

[0006] Ein Nachteil der IDCT-Transformation, so wie dies auch in IEE Std. 1180 bis 1990 dargelegt ist, besteht darin, dass die Berechnung der Transformation eine Matrixmultiplikation von 64-Bit-Gleitkommazahlen beinhaltet, was aus der rechnerischen Perspektive kostenintensiv ist. Dadurch kann die Leistung des Bild- oder Videokompressionssystems, insbesondere beim Streamen von Medien- und medienähnlichen Abspielanwendungen, bei denen das IDCT-Verfahren an großen Mengen von komprimierten Daten auf einer Echtzeit-Basis oder unter anderen zeitähnlichen Einschränkungen durchgeführt wird, gemindert werden.

[0007] Dokument US-A-325 215 schlägt einen Matrixfaktor zum Multiplizieren einer Eingangssignal-Matrix mit einer Koeffizienten-Matrix vor, die so konfiguriert ist, dass sie so viele konstante Faktoren wie absolute Werte der Koeffizienten in der Transformations-Koeffizientenmatrix zum Handlen der Signalmatrix als gemeinsame Multiplikatoren, eine Vielzahl von Selektoren zum Auswählen von Werten, die zum Berechnen der Elemente des Matrixproduktes aus den Multiplikationsergebnissen, welche durch die konstanten Faktoren ausgegeben

werden, notwendig sind und einen jeweilige Akkumulator, der einem jeden der Selektoren zum Akkumulieren der ausgewählten Werte zum abschließenden Erzeugen eines Elementes des Matrixproduktes zugewiesen wird, umfasst.

[0008] Xin Li et al.: "On implementing transforms from integers to integers", Konferenzdokumente der IEE International Conference on Image Processing, Oktober 1998, Seiten 881 bis 885, bezieht sich auf das Ersetzen der linearen Transformationen, die auf Integerwert-Datensequenzen angewendet werden, wodurch nicht-Integer-Ausgänge erzeugt werden, mit nicht-linearen Transformationen, wodurch Integer-Ausgänge erzeugt werden, die mit den Ausgängen entsprechend den linearen Transformationen so nahe wie möglich übereinstimmen.

[0009] Rubino et al.: „Improved Chen-Smith image coder", Konferenzdokumente des IEEE International Symposium on Circuits and Systems, Vol. 2, Mai 1993, Seiten 167 bis 270, beschreibt ein auf Transformationskodierung basierendes Bildkompressionssystem, bei dem die Diskrete Kosinustransformation DCT durch die orthogonale Transformation mit überlappenden Blöcken (LOT [lapped orthogonal transform]) ersetzt wird.

[0010] Dokument US-A-5 590 066 schlägt ein zweidimensionales DCT-System und ein zweidimensionales IDCT-System vor, bei dem beide Elemente zum Multiplizieren, Berechnen und Neuordnen besitzen, wobei jedes dieser Elemente eingerichtet ist, um Daten in serieller Form zu empfangen, um die empfangenen Daten in ihrer seriellen Form zu verarbeiten und die Daten in einer seriellen Form auszugeben, womit Seriell-zu-Parallel-Konverter und Parallel-zu-Seriell-Konverter nicht mehr erforderlich sind.

[0011] „Implementing JPEG with TMS320C2xx Assembly Language Software",TEXAS INSTRUMENT APPLICATION REPORT SPRA615, Januar 2000, Seiten 1 bis 48, ist ein Anwendungsbericht von V. Shao, der sich mit der Implementierung des JPEG-Kompressions-/Dekompressionsverfahrens mit Hilfe der TMS320C2xx Assembly Language Software von Texas Instruments beschäftigt. Im Abschnitt 3.1 wird ein kurzer Überblick über die Diskrete Kosinustransformation zusammen mit einem Beispiel für das Programmieren einer zweidimensionalen 8 × 8-Diskreten Kosinustransformation im Anhang A gegeben. Darüber hinaus werden im Anhang C Tabellen für die Koeffizienten der Diskreten Kosinustransformation aufgeführt, wobei die Koeffizienten durch Integer-Zahlen dargestellt werden.

[0012] „An 8 × 8 Discrete Cosine Transform Implementation on the TMS320C25 or the TMS320C30", TEXAS INSTRUMENT APPLICATION REPORT SPRA115, 1990, Seiten 1 bis 25, ist ein Anwendungsbericht von W. Hohl, der sich mit einer 8 × 8-Diskreten Kosinustransformation-Implementierung mit Hilfe von TMS320C25 oder TMS320C30 beschäftigt. Insbesondere repräsentiert die Gleichung (7) auf Seite 7 ein Erstellungsschema für eine DCT-Matrix.

[0013] Es ist die Aufgabe der vorliegenden Erfindung, ein Verfahren zum Umwandeln eines Mediendatenblocks in einer recheneffizienteren Weise ebenso wie ein entsprechendes Computer-lesbares Medium und einen entsprechenden Konverter bereitzustellen.

[0014] Die Aufgabe wird durch den Gegenstand der unabhängigen Ansprüche erfüllt. Bevorzugte Ausführungsbeispiele der vorliegenden Erfindung sind in den abhängigen Ansprüchen definiert.

[0015] Im Folgenden werden eine Klasse einer ein- und einer zweidimensionalen Transformation, Verfahren zum Erstellen solcher Transformationen und Medienkodierungs- und Mediendekodierungssysteme, die solche Transformationen anwenden, beschrieben.

[0016] Die beschriebenen Transformationen besitzen Implementierungen, die auf Vorgängen der Matrixmultiplikation an Integer-Zahlen zum Erhalten einer Recheneffizienz basieren. In typischen allgemeinen und Grafikprozessoren können Vorgänge der Matrixmultiplikation an Integerzahlen viel schneller als an Gleitkommazahlen durchgeführt werden. Darüber hinaus ermöglichen eine Reihe von 32-Bit-Prozessoren gleichzeitige Multiplikationsoperationen an zwei 16-Bit-Integerzahlen. In einem exemplarischen Ausführungsbeispiel werden die beschriebenen Transformationen mit Matrixmultiplikationen von 16-Bit-Integerzahlen durchgeführt. Die Implementierung der beschriebenen Transformationen mit Integerzahl-Matrixmultiplikation beschleunigt die Durchführung des Kodierens und des Dekodierens in einem Medienkodierungs-/dekodierungssystem.

[0017] Die beschriebenen Transformationen können mit Hilfe einer in diesem Dokument beschriebenen Erstellungsvorgehensweise konstruiert werden. Mit der Erstellungsvorgehensweise werden die Transformationen der Transformationsklasse auf Basis des Auswählens eines Satzes von Transformationskoeffizienten, die

bestimmten Einschränkungen unterliegen, erstellt. Die Einschränkungen können beinhalten, dass die Transformation eine skalierte Integerzahl-Implementierung hat, dass sie eine perfekte oder nahezu perfekte Rekonstruierung gewährleistet, dass sie eine DCT-ähnliche Basis hat, dass sie auf Koeffizienten innerhalb eines Bereiches für die Repräsentation in n-Bits (zum Beispiel ist n 16 Bits) beschränkt ist und Basisfunktionen hat, die in ihren Beträgen nahezu übereinstimmen und dass sie ausreichend Headroom für einen Überlauf des Bereiches bereitstellt.

[0018] Durch Verwenden der beschriebenen Konstruierungsvorgehensweise gibt ein Satz von Transformationen mit 4 und 8-Punkt-Komponenten in einer Dimension Anlass zu 8 × 8-, 8 × 4-, und 4 × 4-Block-Transformationen in zwei Dimensionen. Kompressionssysteme, die auf den Transformationen basieren, die Implementierungen unter Verwendung von 16-Bit- Matrixmultiplikationen zulassen, können ein recheneffizienteres Kodieren und Dekodieren gewährleisten.

[0019] Zusätzliche Leistungsmerkmale und Vorteile der Erfindung werden anhand der folgenden ausführlichen Beschreibung der Ausführungsbeispiele ersichtlich, die zusammen mit den beigefügten Zeichnungen betrachtet werden sollte.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0020] Fig. 1 ist ein Blockdiagramm eines Videokodierers, der auf einer in diesem Dokument beschriebenen Klasse von Transformationen basiert.

[0021] Fig. 2 ist ein Videodekodierer, der auf der beschriebenen Klasse von Transformationen basiert.

[0022] Fig. 3 ist ein Blockdiagramm, dass eine zweidimensionale Transformation darstellt.

[0023] Fig. 4 ist ein Datenflussdiagramm, das die Transformationskodienang unter Verwendung der beschriebenen Klasse von Transformationen mit dem in den Fig. 1 und Fig. 2 dargestellten Videokodierer und Videodekodierer illustriert.

[0024] Fig. 5 ist ein Blockdiagramm einer Implementierung der inversen Transformation für die beschriebene Klasse von Transformationen.

[0025] Fig. 6 ist ein Ablaufplan eines Prozesses zum Erstellen einer der beschriebenen Klasse von Transformationen.

[0026] Fig. 7 ist ein Blockdiagramm einer Implementierung der Vorwärtstransformation für die beschriebene Klasse von Transformationen.

[0027] Fig. 8 ist ein Blockdiagramm einer geeigneten Rechenumgebung für die in den Fig. 1 und Fig. 2 dargestellten Videokodierer/Videodekodierer, die auf der beschriebenen Klasse von Transformationen basieren.

AUSFÜHRLICHE BESCHREIBUNG

[0028] Die folgende Beschreibung konzentriert sich auf eine Klasse von ein- und zweidimensionalen Transformationen zum Erhalten einer Recheneffizienz, auf Verfahren zum Erstelen solcher Transformationen, die bestimmten Kriterien unterliegen, und auf die Verwendung solcher Transformationen bei der Signalverarbeitung, und insbesondere auf Medienkompressionssysteme, die auf solchen Transformationen basieren. Eine exemplarische Anwendung der Transformation ist in einem Bild- oder Videokodierer und- dekodierer enthalten wie beispielsweise einem Kodierer und Dekodierer, der eine Variation des Microsoft Windows Media Video-(WMV) Dateiformats anwendet. Dennoch sind die Transformationen, die wie im Folgenden beschrieben erstellt werden, nicht auf dieses Format beschränkt und können ebenso auf andere Medienkodierungsformate angewendet werden. Dementsprechend werden die Transformationen im Kontext eines allgemeinen Bild- oder Videokodierers und- dekodierers beschrieben, sie können jedoch alternativ dazu in verschiedenen Typen von Mediensignal-Kodierern und -dekodierern eingesetzt werden.

1. Allgemeiner Videokodierer und -dekodierer

[0029] Fig. 1 ist ein Blockdiagramm eines allgemeinen Videokodierers (**100**), und Fig. 2 ist ein Blockdiagramm eines allgemeinen Videodekodierers (**200**).

[0030] Die Beziehungen zwischen den Modulen innerhalb des Kodierers und des Dekodierers zeigen den Hauptfluss von Informtationen in dem Kodierer und dem Dekodierer an; weitere Beziehungen sind aus Gründen der Einfachheit nicht dargestellt. Insbesondere die Fig. 1 und Fig. 2 zeigen für gewöhnlich keine Zusatzinformationen, die die Einstellungen, die Modi, die Tabellen und so weiter, die für eine Videosequenz, einen Frame, einen Makroblock, einen Block und so weiter verwendet werden, anzeigen. Solche Zusatzinformationen werden in dem Ausgangsbitstrom, typischerweise nach dem Entropiekodieren der Zusatzinformationen, gesendet. Das Format des Ausgangs-Bitstroms kann ein Windows Media Video-Format oder ein anderes Format sein.

[0031] Der Kodierer (**100**) und der Dekodierer (**200**) arbeiten auf Block-Basis und verwenden ein 4:2:0-Makroblock-Format, bei dem jeder Makroblock 4 8 × 8-große Luminanzblöcke (mitunter als ein 16 × 16-Makroblock behandelt) und zwei 8 × 8-große Chrominanzblöcke enthält. Alternativ dazu arbeiten der Kodierer (**100**) und der Dekodierer (**200**) auf Objekt-Basis, verwenden ein anderes Makroblock- oder Blockformat, oder führen Operation an Sätzen von Pixeln durch, die eine andere Größe oder Konfiguration als die 8 × 8-Blöcke und die 16 × 16-Makroblöcke haben.

[0032] Je nach Implementierung und Typ von gewünschter Kompression können Module des Kodierers oder des Dekodierers hinzugefügt, weggelassen, in eine Vielzahl von Modulen aufgeteilt, mit anderen Modulen kombiniert und/oder durch gleichartige Module ersetzt werden. In alternativen Ausführungsbeispielen führen Kodierer oder Dekodierer mit anderen Modulen und/oder anderen Modulkonfigurationen ein oder mehrere der beschriebenen Verfahren durch.

A. Videokodierer

[0033] Fig. 1 ist ein Blockdiagramm eines allgemeinen Videokodierersystems (**100**). Das Kodierersystem (**100**) empfängt eine Sequenz von Videoframes einschließlich eines aktuellen Frames (**105**) und erzeugt komprimierte Videoinformationen (**195**) als Ausgang. Bestimmte Ausführungsbeispiele von Videokodierern verwenden typischerweise eine Variation oder eine ergänzte Version des allgemeinen Kodierers (**100**).

[0034] Das Kodierersystem (**100**) komprimiert vorhergesagte Frames und Key-Frames. Im Sinne der geeigneten Darstellung zeigt Fig. 1 einen Pfad für Key-Frames über das Kodierersystem (**100**) und einen Pfad für vorwärts-vorhergesagte Frames. Viele der Komponenten des Kodierersystems (**100**) werden für das Komprimieren von sowohl Key-Frames als auch vorhergesagten Frames verwendet. Die genauen Operationen, die durch diese Komponenten durchgeführt werden, variieren in Abhängigkeit von dem Typ an Informationen, die jeweils komprimiert werden.

[0035] Ein vorhergesagter Frame [auch P-Frame, B-Frame für bidirektionale Vorhersage oder interkodierter Frame genannt] wird hinsichtlich der Vorhersage (oder der Differenz) von einem oder mehreren anderen Frames repräsentiert. Ein Vorhersagerest ist die Differenz zwischen dem, was vorhergesagt worden ist, und dem ursprünglichen Frame. Im Gegensatz dazu wird ein Key-Frame [auch I-Frame, intrakodierter Frame genannt] ohne Bezug auf andere Frames komprimiert.

[0036] Wenn der aktuelle Frame (**105**) ein vorwärts-vorhergesagter Frame ist, sagt ein Bewegungsschätzer (**119**) die Bewegung von Makroblöcken oder anderen Sätzen von Pixeln des aktuellen Frames (**105**) in Bezug auf einen Referenzframe vorher, der der rekonstruierte vorhergehende Frame (**125**), der in dem Frame-Speicher (**120**) zwischengespeichert worden ist, ist. In alternativen Ausführungsbeispielen ist der Referenzframe ein nachfolgend angeordneter Frame, oder der aktuelle Frame wird bidirektional vorhergesagt. Der Bewegungsschätzer (**110**) gibt als Zusatzinformationen Bewegungsinformationen (**115**) wie beispielsweise Bewegungsvektoren aus. Ein Bewegungskompensator (**130**) wendet die Bewegungsinformationen (**115**) auf den rekonstruierten vorhergehenden Frame (**125**) an, um einen bewegungskompensierten aktuellen Frame (**135**) zu erzeugen. Die Vorhersage ist jedoch in den seltensten Fällen perfekt, und die Differenz zwischen dem bewegungskompensierten aktuellen Frame (**135**) und dem ursprünglichen aktuellen Frame (**105**) ist der Vorhersagerest (**145**). Alternativ dazu wenden ein Bewegungsschätzer und ein Bewegungskompensator einen anderen Typ von Bewegungsvorhersage/Bewegungskompensation an. Ein Frequenztransformator (**160**) wandelt die räumlichen Bereichs-Videoinformationen in Frequenzbereich- (das heißt spektrale) Daten um. Für block-basierte Videoframes wendet der Frequenztransformator (**160**) eine Transformation an, die in den folgenden Abschnitten beschrieben wird und die Eigenschaften besitzt, die der Diskreten Kosinustransformation [„DCT"] ähnlich sind. In einigen Ausführungsbeispielen wendet der Frequenztransformator (**160**) eine Frequenztransformation an Blöcken mit räumlichem Vorhersageresten für Key-Frames an. Der Frequenztransformator (**160**) kann eine 8 × 8-, eine 8 × 4-, eine 4 × 8- oder eine Frequenztransformation einer anderen Größe anwenden.

[0037] Ein Quantisierer (**170**) quantisiert anschließend die Blöcke der spektralen Datenkoeffizienten. Der Quantisierer wendet eine einheitliche, skalare Quantisierung auf die Spektraldaten mit einer Schrittgröße an, die auf einer Frame-zu-Frame-Basis oder einer anderen Basis variiert. Alternativ dazu wendet der Quantisierer einen anderen Typ von Quantisierung auf die Spektraldaten-Koeffizienten wie beispielsweise eine nicht-einheitliche Vektor- oder eine nicht-adaptive Quantisierung an, oder er quantisiert direkt räumliche Bereichsdaten in einem Kodierersystem, das keine Frequenztransformationen verwendet. Zusätzlich zu der adaptiven Quantisierung kann der Kodierer (**100**) auch Frame-Dropping, adaptives Filtern oder andere Verfahren zur Ratesteuerung anwenden.

[0038] Wenn ein rekonstruierter Frame für eine anschließende Bewegungsvorhersage/Bewegungskompensation benötigt wird, führt ein inverser Quantisierer (**176**) eine inverse Quantisierung an den quantisierten Spektraldaten-Koeffizienten durch. Anschließend führt ein inverser Frequenztransformator (**166**) die Umkehrung der Operationen des Frequenztransformators (**160**) durch, wodurch ein rekonstruierter Vorhersagerest (für einen vorhergesagten Frame) oder ein rekonstruierter Key-Frame erzeugt wird. Wenn der aktuelle Frame (**105**) ein Key-Frame war, wird der rekonstruierte Key-Frame als der rekonstruierte aktuelle Frame (nicht dargestellt) verwendet. Wenn der aktuelle Frame (**105**) ein vorhergesagter Frame war, wird der rekonstruierte Vorhersagerest zu dem bewegungskompensierten aktuellen Frame (**135**) hinzugefügt, um den rekonstruierten aktuellen Frame zu erzeugen. Der Frame-Speicher (**120**) führt eine Zwischenspeicherung des rekonstruierten aktuellen Frames für die Verwendung in dem nächsten Frame durch. In einigen Ausführungsbeispielen wendet der Kodierer einen deblockierenden Filter auf den rekonstruierten Frame an, um Unregelmäßigkeiten in den Blöcken des Frames adaptiv auszugleichen.

[0039] Der Entropiekodierer (**180**) komprimiert den Ausgang des Quantisierers (**170**) ebenso wie bestimmte Zusatzinformationen (beispielsweise Bewegungsinformationen (**115**), Quantisierungsschrittgröße). Typische Entropiekodierverfahren umfassen das arithmetische Kodieren, Differentialkodieren, die Huffman-Kodierung, die Lauflängenkodierung, LZ-Kodierung, Dictionary-Kodierung sowie Kombinationen der oben genannten Verfahren. Der Entropiekodierer (**180**) verwendet typischerweise verschiedene Kodierungsverfahren für verschiedene Arten von Informationen (beispielsweise DC-Koeffizienten, AC-Koeffizienten, verschiedene Arten von Zusatzinformationen) und kann innerhalb eines bestimmten Kodierverfahrens aus einer Vielzahl von Code-Tabellen auswählen.

[0040] Der Entropiekodierer (**180**) fügt die komprimierten Videoinformationen (**195**) in den Zwischenspeicher (**190**) ein. Ein Zwischenspeicher-Level-Indikator wird zu Bitrate-adaptiven Modulen zurückgemeldet. Die komprimierten Videoinformationen (**195**) werden bei einer konstanten oder relativ konstanten Bitrate aus dem Zwischenspeichert (**190**) entfernt und für das anschließende Streaming bei dieser Bitrate gespeichert. Alternativ dazu führt das Kodierersystem (**100**) ein Streaming der komprimierten Videoinformationen sofort nach der Kompression durch.

[0041] Vor oder nach dem Zwischenspeicher (**190**) können die komprimierten Videoinformationen (**195**) einer Kanalkodierung für die Übertragung über das Netzwerk unterzogen werden. Die Kanalkodierung kann eine Fehlerkorrektur und Korrekturdaten auf die komprimierten Videoinformationen (**195**) anwenden.

B. Videodekodierer

[0042] Fig. 2 ist ein Blockdiagramm eines allgemeinen Videodekodierersystems (**200**). Das Dekodierersystem (**200**) empfängt Informationen (**295**) für eine komprimierte Sequenz von Videoframes und erzeugt einen Ausgang einschließlich einem rekonstruierten Frame (**205**). Bestimmte Ausführungsbeispiele von Videodekodierern verwenden typischerweise eine Variation oder eine ergänzte Version des allgemeinen Dekodierers (**200**).

[0043] Das Dekodierersystem (**200**) dekomprimiert die vorhergesagten Frames und die Key-Frames. Im Sinne der Darstellung zeigt Fig. 2 einen Pfad für Key-Frames über das Dekodierersystem (**200**) und einen Pfad für die vorwärts-vorhergesagten Frames. Viele der Komponenten des Dekodierersystems (**200**) werden für das Komprimieren von sowohl den Key-Frames als auch den vorhergesagten Frames verwendet. Die genauen Operationen, die durch diese Komponenten durchgeführt werden, können in Abhängigkeit von dem Typ von Informationen, die komprimiert werden, variieren.

[0044] Ein Zwischenspeicher (**290**) empfängt die Informationen (**295**) für die komprimierte Videosequenz und stellt die empfangenen Informationen dem Entropiedekodierer (**280**) zur Verfügung. Der Zwischenspeicher (**290**) empfängt typischerweise die Informationen bei einer Rate, die im Verlauf der Zeit ziemlich konstant ist,

und er enthält einen Jitter-Buffer zum Ausgleichen von kurzzeitigen Abweichungen in der Bandbreite oder der Übertragung. Der Zwischenspeicher (**290**) kann einen Abspiel-Zwischenspeicher und andere Zwischenspeicher enthalten. Alternativ dazu empfängt der Zwischenspeicher (**29**) Informationen bei einer sich ändernden Rate. Vor oder nach dem Zwischenspeicher (**290**) können die komprimierten Videoinformationen einer Kanaldekodierung unterzogen werden und hinsichtlich Fehlererfassung und -korrektur verarbeitet werden.

[0045] Der Entropiedekodierer (**280**) führt Entropiekodierung an den entropiekodierten quantisierten Daten ebenso wie an den entropiekodierten Zusatzinformationen (beispielsweise Bewegungsinformationen, Quantisierungsschrittgröße) durch, wobei typischerweise die Umkehrung der in dem Kodierer durchgeführten Entropiekodierung angewendet wird. Entropiedekodierverfahren umfassen das arithmetische Dekodieren, das Differentialkodieren, die Huffman-Dekodierung, die Lauflängendekodierung, die LZ-Dekodierung, die Dictionary-Kodierung sowie Kombinationen der oben genannten Verfahren. Der Entropiedekodierer (**280**) verwendet oftmals verschiedene Dekodierverfahren für die unterschiedlichen Arten von Informationen (beispielsweise DC-Koeffizienten, AC-Koeffizienten, verschiedene Arten von Zusatzinformationen) und kann innerhalb eines bestimmten Dekodierverfahrens aus einer Vielzahl von Code-Tabellen auswählen.

[0046] Wenn der zu rekonstruierende Frame (**205**) ein vorwärts-vorhergesagter Frame ist, wendet ein Bewegungskompensator (**230**) Bewegungsinformationen (**215**) auf einen Referenzframe (**225**) an, um eine Vorhersage (**235**) des Frames (**205**), der rekonstruiert wird, zu erstellen. So verwendet der Bewegungskompensator (**230**) beispielsweise einen Makroblock-Bewegungsvektor, um einen Makroblock in dem Referenzframe (**225**) zu finden. Ein Framespeicher (**220**) speichert die vorhergehend rekonstruierten Frames für die Verwendung als Referenzframes. Alternativ dazu wendet der Bewegungskompensator einen anderen Typ von Bewegungskompensation an. Die Vorhersage durch den Bewegungskompensator ist in den seltensten Fällen perfekt, weshalb der Dekodierer (**200**) auch Vorhersagereste rekonstruiert.

[0047] Wenn der Dekodierer einen rekonstruierten Frame für die anschließende Bewegungskompensation benötigt, führt der Framespeicher (**220**) eine Zwischenspeicherung des rekonstruierten Frames für die Verwendung in der Vorhersage des nächsten Frames durch. In einigen Ausführungsbeispielen wendet der Kodierer einen deblockierenden Filter auf den rekonstruierten Frame an, um Unregelmäßigkeiten in den Blöcken des Frames adaptiv auszugleichen.

[0048] Ein inverser Quantisierer (**270**) führt eine inverse Quantisierung der entropiedekodierten Daten durch. Im Allgemeinen wendet der inverse Quantisierer eine einheitliche skalare Quantisierung an den entropiedekodierten Daten mit einer Schrittgröße an, die auf einer Frame-zu-Frame-Basis oder einer anderen Basis variiert. Alternativ dazu wendet der inverse Quantisierer einen anderen Typ von inverser Quantisierung an den Daten, beispielsweise eine nicht-einheitliche, Vektor- oder nicht adaptive Quantisierung an, oder führt direkt eine inverse Quantisierung der räumlichen Bereichsdaten in dem Dekodierersystem durch, das keine inverse Frequenztransformation verwendet.

[0049] Ein inverser Frequenztransformator (**260**) wandelt die quantisierten Frequenzbereichsdaten in räumliche Bereichs-Videoinformationen um. Für blockbasierte Videoframes wendet der inverse Frequenztransformator (**260**) eine in den folgenden Abschnitten beschriebene inverse Transformation an. In einigen Ausführungsbeispielen wendet der inverse Frequenztransformator (**260**) eine inverse Frequenztransformation an Blöcken mit räumlichen Vorhersageresten für Key-Frames an. Der inverse Frequenztransformator (**260**) kann eine 8 × 8-, eine 8 × 4-, eine 4 × 8- oder eine inverse Frequenztransformation einer anderen Größe anwenden.

II. Transformations-Überblick

[0050] Fig. 3 illustriert eine zweidimensionale Transformation **300** und eine inverse Transformation **310**, die in dem in den Fig. 1 und Fig. 2 dargestellten Videokodierer und Videodekodierer verwendet werden. Die Transformation **300** und die inverse Transformation **310** sind von einer Klasse von Transformationen, die, wie im Folgenden beschrieben, bei ihrer Erstellung bestimmten Einschränkungen unterliegen.

[0051] Die zweidimensionale Transformation **300** wandelt einen zweidimensionalen (n × m) Block **320** mit Medieninhalt, der durch räumlich-bezogene Samples des Medieninhaltes repräsentiert wird, in einen Transformationsbereichs-Block um. So kann der Block beispielsweise ein Abschnitt eines digitalen Bild- oder eines digitalen Videoframes sein, wobei ein solcher als eine Gruppe von Farbsamples (Pixel), die in einheitlich zueinander beabstandeten Gitter-Orten positioniert sind, repräsentiert sein kann. Dies wird als räumliche Bereichs-Repräsentation des Medieninhaltes bezeichnet. Der Transformationsblock ist darüber hinaus auch aus n × m Samplen, und zwar in einer, was in diesem Dokument als Transformationsbereichs-Repräsentation bezeichnet wird,

gebildet.

[0052] Die inverse Transformation **320** wandelt einen Block von Samples von dem Transformationsbereich zurück in den ursprünglichen oder den räumlichen Bereich um.

III. Transformations-basiertes Kodieren

[0053] Im Allgemeinen verwendet das Transformations-basierte Kodieren **400** von Medieninhalt, wie beispielsweise in dem Videokodierer und -dekodierer wie oben beschrieben, die Transformation **300** (Fig. 3) und die inverse Transformation **310** (Fig. 3) zusammen mit der Quantisierung, um den Medieninhalt zu einer komprimierten Form zu kodieren. Zunächst wendet die Transformations-basierte Kodierung die Transformation **300** in einer Transformationsstufe **410** auf einen Eingangsblock des Medieninhaltes für die Umwandlung zu einem Transformationsbereich an. Anschließend führt die Transformations-basierte Kodierung Quantisierung (das heißt, reduziert die Auflösung) von bestimmten Transformationsbereichs-Samplen (beispielsweise die, die eine weniger wahrnehmbare Verschlechterung des Medieninhaltes aufweisen) in einer Quantisierungsstufe **420** durch. Die quantisierten Transformationsbereichs-Samples können verwendet werden, um die komprimierte Form des Medieninhaltes zu erzeugen.

[0054] Die Transformations-basierte Kodierung **400** weist darüber hinaus auch eine inverse Quantisierungsstufe **430** und eine inverse Transformationsstufe **440** auf. In der inversen Quantisierungsstufe **430** führt die Transformations-basierte Kodierung ein Mapping der quantisierten Transformationsbereichs-Samples zurück zu ihrer ursprünglichen Auflösung in Vorbereitung auf die inverse Transformation **310** durch. Die Transformations-basierte Kodierung führt in der inversen Transformationsstufe inverse Transformation an den dequantisierten Transformationsbereichs-Samplen durch, um anschließend den Medieninhaltblock zu rekonstruieren.

[0055] Die Transformations-basierte Kodierung **400** kann an verschiedenen Stellen in dem Videokodierer- und dekodierer durchgeführt werden. So kann der Videokodierer beispielsweise auch eine Rekonstruierungsschleife mit den Stufen der inversen Quantisierung und der inversen Transformation enthalten, die bei den Verfahren der Differenzial-Kodierung und der Interframe-Kodierung verwendet wird.

III. Recheneffiziente Implementierung der inversen Transformation

[0056] Im Folgenden sind in Bezug auf Fig. 5 die Transformation **300** und die inverse Transformation **310** (Fig. 3) vorzugsweise als eine Vorher-Multiplikation **510** des zweidimensionalen Datenblockes (aus räumlichen Bereichssamplen für die Transformation **300**, und aus Transformationsbereichs-Samplen für die inverse Transformation) durch eine Vorher-Multiplikationsmatrix (T) und als eine Nachher-Multiplikation **530** durch eine Nachher-Multiplikationsmatrix (T') implementiert. Die Zeilen der Vorher-Multiplikationsmatrix (T) repräsentieren die Basisfunktionen der Transformation, die in der Vorher-Multiplikation **510** auf die Spalten des Datenblockes angewendet werden. Auf ähnliche Weise sind die Spalten der Nachher-Multiplikationsmatrix (T') die Basisfunktionen der Transformation, die auf die Zeilen des Datenblockes in der Nachher-Multiplikation **530** angewendet werden.

[0057] Um eine Recheneffizienz zu erhalten, sind die Transformationsmatrizen (T und T') sowie der Datenblock aus Integerzahlen innerhalb von Bereichen gebildet, die zulassen, dass die Matrixmultiplikationen unter Verwendung der Integer-Multiplikationsoperationen des Computer- oder des Grafikprozessors durchgeführt werden können. So ist es beispielsweise mit vielen aktuell verwendeten Prozessoren, die 16-Bit-Integer-Multiplikationsoperationen bereitstellen, der Fall, dass die Matrizen vorzugsweise aus Integerzahlen innerhalb eines Bereiches gebildet sind, der zulässt, dass die Matrixmultiplikationen unter Verwendung der 16-Bit-Integer-Multiplikationsoperationen durchgeführt werden können. Alternativ dazu können die Matrizen aus Integerzahlen in kleineren oder größeren Bereichen für die Prozessoren gebildet sein, die Integer-Multiplikationsoperationen von Integerzahlen anderer Größenordnungen bereitstellen.

[0058] Durch die Vorher-Multiplikation und die Nachher-Multiplikation des Datenblockes durch die Basisfunktionen der Transformation, die aus Integerzahlen gebildet sind, werden die resultierenden Datenblockwerte in einem größeren Bereich erzeugt. Die Implementierung **500** kompensiert diese Vergrößerung durch die Basisfunktions-Multiplikation, die die Skalierungsoperationen **520**, **540** jeweils nach der Vorher-Multiplikation und nach der Nachher-Multiplikation verwendet. Um eine Recheneffizienz zu erhalten, sind die Skalierungsoperationen **520**, **540** vorzugsweise Verschiebungsoperationen (die eine Teilung durch eine Zweierpotenz bewirken), die jeweils die Werte einer Anzahl von Bitpositionen S_1 und S_2 verschieben.

[0059] In der illustrierten Implementierung **500** sind die Werte in dem Datenblock Integerzahlen, die jeweils beim Eingang und dem Ausgang der Vorher-Multiplikation **510**, der Skalierung **520**, der Nachher-Multiplikation **530**, der Skalierung **540** Bitgrößen haben, die jeweils als A bis E repräsentiert werden. So sind beispielsweise die Werte des Datenblockes an dem Eingang der Vorher-Multiplikationsstufe **510** Integerzahlen, die eine Größe von A-Bits aufweisen.

[0060] Es ist ein zugrundeliegendes Prinzip in der Ausführung dieser Transformation, dass ihre Implementierung **500** als Paar aus Vorwärts- und inverser Transformation ausgeführt ist, wobei die Letztere in Integer-Arithmetik beschränkter Präzision implementiert ist, bei der die inverse Transformation sicherstellt, dass ein bedeutungsvolles Ergebnis für die Eingangsdaten erzeugt wird, die durch den entsprechenden Vorwärtstransformationsprozess (unterliegt gültiger Quantisierung und Dequantisierung) erzeugt worden sind.

IV. Transformations-Erstellung

[0061] Die recheneffizienten Transformationen werden durch Auswählen von Koeffizientenwerten für die Basisfunktionen der Transformation (das heißt die Werte in der Vorher-Multiplikationsmatrix T und in der Nachher-Multiplikationsmatrix T'), die bestimmten Beschränkungen unterliegen, welche im Folgenden beschrieben werden, erstellt. Dieses Konstruierungsverfahren kann eine Reihe von beschränkten, orthogonalen oder biorthogonalen Transformationen hervorbringen.

Beschränkungen.

[0062] Skalierte Integer-Implementierung. Die Transformationskoeffizienten sind Integerzahlen mit einer möglichen Skalierung durch eine Zweierpotenz. Dadurch wird die Implementierung auf einem Standardcomputer erleichtert.

[0063] Perfekte Rekonstruierung. Bei Nicht-Vorhandensein von Quantisierung (beispielsweise der in **Fig. 4** dargestellten Quantisierungs- und inversen Quantisierungsstufe **420**, **430**) führt die inverse Transformation eine perfekte Rekonstruierung der ursprünglichen räumlichen Bereichsdaten aus den durch die Transformation erzeugten Transformationsbereichsdaten durch. Die erste Anforderung für eine Transformation, die zum Komprimieren von Daten verwendet wird, besteht darin, dass die Vorwärts- und die inverse Transformation bei Nicht-Vorhandensein von Quantisierung und Runden ein Paar für eine perfekte Rekonstruierung bilden. Dies kann durch Orthonormalität oder Bi-Orthonormalität gewährleistet werden. In dem ersteren Fall sind die Vorwärts- und die inverse Transformation einander identisch, wobei sie sich in dem letzteren Fall voneinander unterscheiden.

[0064] DCT-ähnliche Basis. Die Transformation und die inverse Transformation besitzen Eigenschaften, die der Diskreten Kosinustransformation (DCT) ähnlich sind. Um eine gute Leistung beim Kodieren zu erhalten, wird es bevorzugt, dass die Transformation mit der DCT annähernd übereinstimmt. Die DCT ist dafür bekannt, dass sie günstige Energieverdichtungseigenschaften besitzt, die in einer nahezu optimalen Datenkompression resultieren. Die DC-Basisfunktion der DCT ist ein konstanter Wert. Dies beschränkt die Transformationskoeffizienten der Nullten" Basis dahingehend, dass sie konstante Werte besitzen.

[0065] Bereich. Der Bereich der Datenwerte und der Transformationskoeffizienten ermöglicht eine Implementierung unter Verwendung von recheneffizienten Integer-Multiplikationsoperationen auf dem Zielcomputer- oder Grafikprozessor (beispielsweise in 16-Bit-Integer-Operationen). In einem bevorzugten Ausführungsbeispiel, sind die Werte auf einen Bereich beschränkt, der eine 16-Bit-Implementierung der inversen Transformation ermöglicht. In dieser Implementierung ist die 16-Bit-Integer-Operation eine arithmetische Operation an Integerzahlen, die mit einem 16-Bit-Akkumulator durchgeführt werden kann und die Modulo- oder Rollover-arithmetisch auf die Basis 2^{16} signiert werden kann, das heißt ein Akkumulator mit einem Datenbereich von [–32768 ... 32767]. Die Multiplikation von zwei 16-Bit-Zahlen resultiert in lediglich den niederwertigen 16 Bits des erhaltenen Produkts. Alternativ dazu kann der Bereich für andere Zielplattformen, die eine andere Bitgröße der Integer-Operationen unterstützen, variieren.

[0066] Die Multiplikation der Samples in dem Datenblock durch die Transformations-Basisfunktionen bewirkt eine Erweiterung des Bereichs, der mit der Anzahl von Samplen (Punkten) in der Transformations-Basisfunktion variiert. In Videokompressionssystemen werden zweidimensionale (N × M-Punkte) Transformationen von 8 × 8-, 4 × 8-, 8 × 4- und 4 × 4-Punkten gemeinsam verwendet. Zwischen der 4- und der 8-Punkt-Transformation, platziert die 8-Punkt-Transformation die strikteren Beschränkungen auf den Bereich der Integer-Transformationskoeffizienten. Dies geschieht deshalb, da die mit der N-Punkt-Transformation assoziierte Erweiterung

größer als die mit der M-Punkt-Transformation assoziierte Erweiterung ist, wenn N > M ist. Der DC-Wert erweitert sich als √N für eine N-Punkt-Transformation. Dementsprechend betrachten wir zunächst die 8-Punkt-Transformation, da sie die strikteren Beschränkungen repräsentiert.

[0067] So ist beispielsweise ein gemeinsamer Bereich für die Samples des räumlichen Bereichsdatenblocks in Videokompressionssystemen [–255 ... 255], von dem die Auflösung 9-Bits ist. Für solche 9 Bits, die entsprechend einem Bereich von [–255 ... 255] eingegeben werden, kann die 8 × 8-Transformation Werte in dem Bereich von [–2047 2047] annehmen, wobei eine Auflösung von 12 Bits erforderlich wird. Tatsächlich resultiert jede 8-Punkt-Transformation in einer Erweiterung von 1,5 Bits. Für eine zweidimensionale Transformation wird eine 8-Punkt-Transformation zwei Mal angewendet, wobei Zeilen-Transformationen in der Vorher-Multiplikationsstufe **510** (Fig. 5) und die Spalten-Transformationen in der Nachher-Multiplikationsstufe **530** (Fig. 5) so durchgeführt werden, dass die resultierenden Transformationsbereichs-Samples durch 3 Bits erweitert werden (zu einer 12-Bit-Auflösung).

[0068] Anhand eines noch spezielleren Bezuges auf Fig. 5 wird ersichtlich, dass der Eingangs-Transformationsbereichs-Datenblock beim Eingang A für die inverse Transformation in einem exemplarischen Ausführungsbeispiel einen Bereich von 12 Bits hat, wobei der rekonstruierte räumliche Bereichsdaten-Block beim Ausgang E einen Bereich von 9 Bits hat. Zwei inverse Transformationsoperationen (Vorher-Multiplikation **510** und Nachher-Multiplikation **530**) veranlassen eine Skalierung oder eine Bereichserweiterung von ITI^2, die durch zwei Verschiebungen von jeweils s_1-Bits und s_2-Bits kompensiert wird. Dies erfordert, dass die Bereichserweiterung, die durch die Transformationen veranlasst wird, der ausgleichenden Skalierung annähernd gleich ist, beziehungsweise, dass $ITI^2 \approx 2(s_1 + s_2)$ ist. Angenommen, die Transformationsmatrix (T) ist eine normalisierte Matrix (bei der $T_1 = T/ITI$), dann liegt der dynamische Bereich des resultierenden Datenblockes (B = T × A) nach der Vorher-Multiplikation ungefähr bei 10,5 Bits. Der resultierende Datenblock (B) hat demzufolge einen Bereich von $10,5 + \log_2(ITI)$ Bits. Auf die gleiche Weise besitzt der Datenblock (D = C × T') der Nachher-Multiplikation einen Bereich von $9 + 2 \times \log_2(ITI) - s_1$ Bits. Um eine recheneffiziente Implementierung der inversen Transformation unter Verwendung von 16-Bits-Integer-Operationen zu erhalten, sind die folgenden Relationen erforderlich (wobei $\log_2(ITI)$ mit L bezeichnet wird):

$$10,5 + L <= 16 \quad (1)$$

$$9 + 2L - s_1 <= 16 \quad (2)$$

$$2L \approx s_1 + s_2 \quad (3)$$

[0069] Daraus ergibt sich, dass (9 + m <= 16), das heißt m = 6; und L <= 5,5 oder ITI <= 2048 ist.

[0070] Wenn darüber hinaus die DC-Basisfunktion der inversen Transformation durch [d d d d d d d d] dargestellt ist, erfordert die Bereichsbeschränkung, dass $8\,d^2 <= 2048$, oder d <= 16 ist.

[0071] Beträge der Basisfunktionen. Die Beträge der Basisfunktionen für die Transformation stimmen nahezu miteinander überein.

[0072] In allgemeinen orthogonalen und bi-orthogonalen Integer-Transformationen ist es zugelassen, dass unterschiedliche Zeilen der Transformationsmatrix T entsprechend unterschiedlichen Basisfunktionen unterschiedliche Beträge haben. Der Gedanke hinter dieser Flexibilität ist der, dass die Auswirkung einer Ungleichheit zwischen den Beträgen während der Quantisierung und der inversen Quantisierung für die Vorwärts- und die inverse Transformation annulliert werden kann. In der Praxis stellt die Normalisierung zwei Schwierigkeiten dar – (1) durch Normalisierung wird insbesondere die Dekodierer-Seite komplexer und (2) es sei denn, die Normalisierungs-Faktoren sind kleine Integerzahlen, wird die Normalisierung für eine kurze Integerzahl- (16 Bit) Implementierung unmöglich.

[0073] Wenn es erforderlich ist, dass die Beträge aller Basisfunktionen übereinstimmen, wird eine zusätzliche erneute Normalisierung nicht notwendig. Eine beliebige Normalisierungs-Bedingung kann in eine inverse Quantisierung gebracht werden. Die Auswirkung von Quantisierung von Werten kann in diesem Prozess ein wenig verschoben werden (das heißt der Rate-Distortion-Punkt eines bestimmten Quantisierungspunktes (QP) kann sich weg bewegen, jedoch entlang der Rate-Distortion-(R-D) Kurve von seinem ursprünglich berechneten Wert, das heißt unter Verwendung einer normalisierten Transformation absoluter Präzision).

[0074] Wird eine 4- und 8-Punkt-Transformation durchgeführt, trifft dieselbe Logik zu. Es kann erforderlich

sein, dass die Beträge aller Basisfunktionen, egal ob 4- oder 8-Punkt, übereinstimmen müssen, so dass die Normalisierung wegfallen kann. In der Praxis ist es für beachtlich kleine Integer-Basen unmöglich, einen Satz von Transformationskoeffizienten zu finden, der einer Beschränkung unterliegt, bei der alle Beträge sämtlicher Basisfunktionen übereinstimmen. Die Beschränkung hierbei lässt einen geringen Grad von Flexibilität durch Zulassen von nur denjenigen Basisfunktionen zu, von denen die Beträge annähernd übereinstimmen.

[0075] Headroom. Obgleich in einer Implementierung der Bereich des gültigem Eingangs (der räumliche-Bereichsdatenblock) zu der Vorwärtstransformation bei 9 Bits liegt, besteht die Möglichkeit, dass der rekonstruierte Datenblock nach der Quantisierung (Stufe **420** in Fig. 4) einen Überlauf über oder unter den Bereich +255 bis –255 erfährt. Die Implementierung der kurzen Integerzahl muss dies notwendigerweise berücksichtigen, indem sie ausreichend Headroom für solche Situationen verfügbar hat.

[0076] Zusammenfassend erstellt die in diesem Dokument beschriebene Konstruierungsvorgehensweise Transformationen, die den folgenden Beschränkungen unterliegen: (1) skalierte Integer-Implementierung, (2) Orthonormalität oder perfekte Rekonstruierung; (3) DCT-ähnliche Basis für Energieverdichtung und (4) beschränkter Bereich für eine Integer-basierte Implementierung (beispielsweise in 16-Bit-Integer-Operationen). Für einen jeweils gegebene Bereichsbeschränkung erweist es sich als schwierig, alle der oben genannten Beschränkungen zu erfüllen. Als ein bestimmtes Beispiel, ist es nicht möglich, eine solche Transformation zu erstellen, die die ersten drei der genannten Beschränkungen erfüllt und darüber hinaus auch noch eine Bereichsbeschränkung auf 16 Bits erfüllen soll. Mit dem im Folgenden beschriebenen Konstruierungsprozess werden effiziente, Bereichs-beschränkte Transformationen durch ein leichtes Lockern einer oder mehrerer Beschränkungen (beispielsweise durch Lockern der Orthonormalitäts-Beschränkung in der illustrierten Transformations-Implementierung) erzeugt. In der dargestellten Transformations-Implementierung wird von dem Transformations-Paar lediglich erfordert, dass es orthogonal ist, währenddessen die Beträge leicht voneinander abweichen können. Diese zusätzliche Flexibilität erweitert den Suchbereich für Transformationen, wodurch das Konstruieren effizienter Bereichs-beschränkter Transformationen machbar wird, die wiederum die anderen Beschränkungen erfüllen. Wie im weiteren Verlauf diskutiert wird, kann die Auswirkung der leichten Ungleichheit in den Beträgen durch eine erneute Normalisierung während der Quantisierungsstufe annulliert werden. Mit der erneuten Normalisierung, die in der Quantisierungsstufe implementiert wird, wird die resultierende Zunahme an Komplexität lediglich im Kodierer bewirkt. Dadurch wird die Komplexität des gesamten Systems insgesamt erheblich reduziert, währenddessen die Effizienz bei der Kompression aufrechterhalten wird.

Konstruierung.

[0077] Im Folgenden wird in Bezug auf Fig. 6 ein Prozess **600** zum Konstruieren einer recheneffizienten Transformation, die diesen Beschränkungen unterliegt, ersichtlich, und der einen Satz von orthogonalen Transformationskoeffizienten für die Matrizen der Vorher-Multiplikation und der Nachher-Multiplikation (T und T') für die in Fig. 5 dargestellte Transformations-Implementierung erzeugt. Einige der Toleranzschwellenwerte, die in dieser Konstruierungs-Vorgehensweise zum Erstellen dieser exemplarischen recheneffizienten Transformations-Implementierung verwendet werden, sind beliebig gewählt und können noch weiter gelockert werden, um weitere Lösungen zu erzielen. Obgleich festzuhalten gilt, dass aufgrund solcher gelockerten Schwellenwerte solche alternativen Transformations-Implementierungen eine schlechtere Kodierungs-Leistung aufweisen können.

[0078] Ein exemplarischer Satz von recheneffizienten Transformationen, der unter Verwendung dieses Prozesses **600** konstruiert wird, enthält zweidimensionale Transformationen, die 4 und 8-Punkt-Basisfunktionen verwenden. Auf Basis der 4- und 8-Punkt-Basisfunktionen werden Transformationsmatrizen für 8 × 8-, 4 × 8-, 8 × 4- und 4 × 4-Transformationen erzeugt. Alternativ dazu kann der Konstruierungsprozess jedoch auch so variiert werden, dass Transformationen mit Basisfunktionen erzeugt werden, die eine andere Anzahl von Punkten oder Blockdimensionen aufweisen.

[0079] Ein erster Schritt **610** in dem Konstruierungsprozess **600** besteht darin, den konstanten Faktor der DC-Basisfunktion zu finden. Für die 4-Punkt- und die 8-Punkt-Basisfunktionen werden die konstanten Faktoren jeweils mit d_4 und d_8 bezeichnet. Gemäß der Beträge-Beschränkung, die voranstehend diskutiert wurde, werden diese konstanten Faktoren in der folgenden Relation dargestellt: $d_4 \approx \sqrt{2}d_8$, so dass $d_8 \leq 16$ ist. Die einzigen Integer-Paare $\{d_4, d_8\}$, die diese Beträge-Beschränkung innerhalb von ungefähr 1 % erfüllen sind {7, 5}, {10, 7}, {17, 12} und {20, 14}. Für die 8-Punkt-Transformation ist der Betrag im Quadrat der DC-Basis $8d_8^2$. Die zulässigen 8-Punkt-Beträge im Quadrat für die DC-Basis mit diesen Integer-Paaren sind demzufolge 200, 392, 1152 und 1568.

[0080] Ein zweiter Schritt **620** in dem Konstruierungsprozess **600** besteht darin, die ungeraden Basisfunktionen (auch als die ungeraden „Frequenzen" bezeichnet) der Transformation zu bestimmen. In Übereinstimmung mit der „DCT-ähnlichen Basis"-Beschränkung, die voranstehend diskutiert wurde, sollten die Transformationen Eigenschaften haben, die der DCT-Transformation ähnlich sind, die für die 8-Punkt-DCT-Transformation die folgenden ungeraden Basisfunktionen hat:

$$\begin{bmatrix} 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{bmatrix}$$

[0081] Vier konstante Faktoren bestimmen diese ungeraden Basisfunktionen der 8-Punkt-DCT. Aufgrund der Struktur der Basen können die Faktoren C1, C2, C3 und C4 zum Ersetzen der eindeutigen absoluten Koeffizienten verwendet werden, wodurch die folgende Basis entsteht:

$$\begin{bmatrix} C1 & C2 & C3 & C4 & -C4 & -C3 & -C2 & -C1 \\ C2 & -C4 & -C1 & -C3 & C3 & C1 & C4 & -C2 \\ C3 & -C1 & C4 & C2 & -C2 & -C4 & C1 & -C3 \\ C4 & -C3 & C2 & -C1 & C1 & -C2 & C3 & -C4 \end{bmatrix}$$

[0082] Die Konstruierung fährt mit dem Durchsuchen des Raumes $\{C_1, C_2, C_3, C_4\}$ fort, wobei die folgenden Bedingungen gelten:

1. Orthogonalität der Basis. Orthogonalität der ungeraden Bedingungen mit geraden Frequenzbedingungen ist in Anbetracht der komplementären Struktur gerader Frequenzen implizit. Aus diesem Grund reduziert sich diese Bedingung auf Orthogonalität der ungeraden Basisfunktionen.
2. DCT-ähnliche Basis. Der Vektor $[C_1, C_2, C_3, C_4]$ korreliert „gut" mit dem entsprechenden DCT-Koeffizienten-Vektor [0,4904 0,4157 0,2778 0,0975].
   Die Korrelation wird durch den Kosinus des Winkels zwischen den Vektoren gemessen und ist wünschenswerterweise so nahe wie möglich bei 1. Es können ebenso andere Messungen für die Korrelation verwendet werden.
3. Die Beträge der ungeraden Basen stimmen annähernd mit dem DC-Betrag überein. Dies kann wie folgt ausgedrückt werden:

$C_1^2 + C_2^2 + C_3^2 + C_4^2 \approx$ einer von {100, 196, 576, 784}

[0083] Innerhalb einer Toleranz von 5 % des Betrages (Bedingung 3) und der zulässigen Korrelation „Kosinus" von 0,99 (Bedingung 2) ist nur ein Satz von Integerkoeffizienten vorhanden, der die Beschränkungen für die ungeraden 8-Punkt-Basisfunktionen erfüllt.

[0084] Dieser Satz ist [16 15 9 4] und hat einen Betrag von 578, der definitiv nahe bei dem gewünschten Betrag von 576 liegt. Die Korrelation ist eine günstige und beträgt 0,9984.

[0085] Ein dritter Schritt **630** des Konstruierungsprozesses besteht darin, die geraden Basisfunktionen oder die geraden Frequenzen für die Transformation zu bestimmen. Erneut sollten gemäß der Beschränkung der DCT-ähnlichen Basis die geraden Basisfunktionen gut mit denen der DCT-Transformation korrelieren. Für die 8-Punkt-DCT-Transformation werden die geraden Basisfunktionen wie folgt ausgedrückt:

$$\begin{bmatrix} 12 & 12 & 12 & 12 & 12 & 12 & 12 & 12 \\ C5 & C6 & -C6 & -C5 & -C5 & -C6 & C6 & C5 \\ 12 & -12 & -12 & 12 & 12 & -12 & -12 & 12 \\ C6 & -C5 & C5 & -C6 & -C6 & C5 & -C5 & C6 \end{bmatrix}$$

[0086] Die Betragsbeschränkung auf den Frequenzen 2 und 6 wird durch $C_5^2 + C_6^2 \approx 288$ ausgedrückt. Das Integerpaar {16, 6} erfüllt diese Betragsbeschränkung innerhalb von ungefähr 1 %.

[0087] Die resultierende 8-Punkt-Transformationsmatrix (unter Verwendung der in den Schritten 2 und 3 bestimmten Integerkoeffizienten) ist die folgende:

$$T_8 = \begin{bmatrix} 12 & 12 & 12 & 12 & 12 & 12 & 12 & 12 \\ 16 & 15 & 9 & 4 & -4 & -9 & -15 & -16 \\ 16 & 6 & -6 & -16 & -16 & -6 & 6 & 16 \\ 15 & -4 & -16 & -9 & 9 & 16 & 4 & -15 \\ 12 & -12 & -12 & 12 & 12 & -12 & -12 & 12 \\ 9 & -16 & 4 & 15 & -15 & -4 & 16 & -9 \\ 6 & -16 & 16 & -6 & -6 & 16 & -16 & 6 \\ 4 & -9 & 15 & -16 & 16 & -15 & 9 & -4 \end{bmatrix}$$

[0088] Ein vierter Schritt **640** des Konstruierungsprozesses **600** erzeugt die 4-Punkt-Transformation. Für die 8-Punkt-Transformation, die wie oben beschrieben erstellt wurde (Schritte **610** bis **630**), ist der konstante Faktor der DC-Basisfunktion de = 12. Der konstante Faktor für die 4-Punkt-Transformation ist demzufolge $d_4 = 17$. Für eine DCT-ähnliche Basis ergibt sich, dass die Koeffizienten D_1 und D_2 in den folgenden Basisfunktionen zu erzeugen sind:

$$\begin{bmatrix} 17 & 17 & 17 & 17 \\ D1 & D2 & -D2 & -D1 \\ 17 & -17 & -17 & 17 \\ D2 & -D1 & D1 & -D2 \end{bmatrix}$$

[0089] Die Matrix mit oben beschriebenen Basisfunktionen ist inhärent orthogonal für eine beliebige Auswahl von D_1 und D_2. Gemäß der Beschränkung der DCT-ähnlichen Basis werden die Koeffizienten jedoch durch den DCT-Twiddle-Factor ins Verhältnis gesetzt (das heißt das Verhältnis D_1/D_2), wobei dieser $\sqrt{2} + 1$ entspricht. Darüber hinaus unterliegen die Koeffizienten der Betragsbeschränkung $D1^2 + D2^2 \approx 578$. Ein Integerpaar, das die Beschränkung der Orthogonalität, die Beschränkung der DCT-ähnlichen Basis und die Beschränkung des Betrages innerhalb bis zu ungefähr 1 % erfüllt, ist das Integerpaar {22, 10}. Daraus ergeben sich die folgenden Basisfunktionen für die 4-Punkt-Transformation:

$$T_4 = \begin{bmatrix} 17 & 17 & 17 & 17 \\ 22 & 10 & -10 & -22 \\ 17 & -17 & -17 & 17 \\ 10 & -22 & 22 & -10 \end{bmatrix}$$

V. Transformations-Implementierung (Fortsetzung)

[0090] Unter nochmaliger Bezugnahme auf **Fig. 5** definieren die Transformationsmatrizen, die unter Verwendung des oben beschrieben Konstruierungsprozesses erzeugt werden, die Transformationen maßgeblich. Für eine zweidimensionale Transformation werden die Basisfunktionen in jeder Dimension auf den Sample-Datenblock angewendet. In der illustrierten Transformations-Implementierung **500** werden die Basisfunktionen in der Vorher-Multiplikation **510** auf Zeilen des Sample-Datenblockes angewendet und anschließend in der Nachher-Multiplikation **530** auf Spalten des Sample-Datenblockes. Da die Transformation so konstruiert ist, dass sie orthogonal ist, werden die identischen Transformationsmatrizen für die Vorwärts- **300** und für die inverse Transformation **310** verwendet. Mit den Transformations-Basisfunktionen mit nicht-einheitlichen Beträgen, enthält die illustrierte Transformations-Implementierung darüber hinaus auch Skalierungsstufen **520** und **540** bei der inversen Transformation, um die Bereichserweiterung zu kompensieren, die durch die Transformationen verursacht worden ist. Solch ein Bereichs-beschränkende Transformation kann auch für die Vorwärtstransformation verwendet werden, jedoch ist im Kontext des Videokodierens das Beschränken des Bereiches bei der Vorwärtstransformation oftmals redundant. Da der Kodierungsprozess langsamer von statten geht als der Dekodierungsprozess, und da die verfügbaren Rechenressourcen in dem Kodierer oftmals eine Größenordnung höher als die im Dekodierer sind, kann der Kodierer im höherem Präzisions- (beispielsweise 32 Bit) Integer oder sogar doppeltem Präzisions-Gleitkomma implementiert sein. Dementsprechend betrachtet die folgende Diskussion zunächst eine exemplarische Implementierung der inversen Transformation unter Verwendung der oben beschriebenen Transformationsmatrizen.

Implementierung der Inversen Transformation

[0091] Bei der Transformations-Implementierung **500**, werden, wie voranstehend beschrieben wurde, die Zei-

len des Transformationsbereichs-Datenblockes (den wir in der folgenden Diskussion als D bezeichnen) in der Vorher-Multiplikationsstufe **510** inverser Transformation unterzogen, indem eine Matrixmultiplikation mit der Transformationsmatrix der geeigneten Größe (beispielsweise die oben stehende Matrix T_8 für einen 8 × 8-Datenblock) durchgeführt wird. Die Spalten des Transformationsbereichs-Datenblockes werden in der Nachher-Multiplikationsstufe **530** durch eine Matrixmultiplikation ebenfalls mit der Transformationsmatrix inverser Transformation unterzogen. Diese Reihenfolge kann zwar umgekehrt werden, würde jedoch dann zu einer Ungleichheit führen. Die Transformationsmatrix für die Vorher-Multiplikations- und die Nachher-Multiplikationsstufen ist die, die wie oben beschrieben konstruiert wurde.

[0092] Für die Skalierungsstufen **520** und **540** wird das Skalieren in der exemplarischen Transformations-Implementierung bestimmt, indem es der oben beschriebenen Beschränkung der skalierten Integerimplementierung unterliegt (das heißt, die Skalierung erfolgt bei einer Zweierpotenz, um die Berechnung auf einem Standardsystem- und einem Grafikprozessor zu ermöglichen). Dementsprechend wird die Skalierung als die Zweierpotenz ausgewählt, die dem Betrag im Quadrat der Basisfunktion der Transformationsmatrix am nächsten ist.

[0093] Genauer gesagt, kann die vorschriftsmäßige Formel für die inverse Transformation wie folgt dargestellt werden:

$$R = \frac{(T_8' \cdot D \cdot T_8)}{s}$$

wobei D den Transformationsbereichs-Datenblock an dem Eingang (A) zu der inversen Transformation bezeichnet. In der folgenden Diskussion stellt D_1 den Datenblock-Ausgang von der ersten Multiplikationsstufe **510** der Transformations-Implementierung **500** dar, und R stellt den rekonstruierten Ausgang nach der zeilen- und spaltenweisen inversen Transformation dar. Der Nenner s ist der Skalierungsfaktor. D, D_1 und R sind isomorphische 8 × 8-, 8 × 4- oder 4 × 8-Matrizen. Entgegen der üblichen mathematischen Darstellungsart sind die Operationen, die eine Matrix und einen Skalar verwenden, einträgeweise Operationen an der Matrix sind. Auf die gleiche Weise sind skalare Operationen mit einem Matrixargument einträgeweise skalare Operationen an der Matrix.

[0094] Der Nenner s wird als die Zweierpotenz ausgewählt, die dem Betrag im Quadrat der Basisfunktionen am nächsten ist. Für die 8 × 8-Transformation sind die Werte der Beträge im Quadrat der eindimensionalen 8-Punkt-Basisfunktionen {1152, 1156, 1168}. Aus diesem Grund wird der Nenner mit 1024 (das heißt, s = 1024) ausgewählt, der die Zweierpotenz ist, die diesen Werten der Beträge im Quadrat am nächsten ist. Da das Verhältnis zwischen den eigentlichen Beträgen und diesem Nenner (das heißt, Beträge/s ≈ 1,12) nahe bei 1 ist, besteht eine enge Übereinstimmung zwischen dem Quantisierungsparameter, der für die Standard-IDCT und dem, der für die illustrierte Transformations-Implementierung verwendet wird. Hierbei wird kein zusätzlicher Fehler hervorgerufen, da die gesamte verbleibende Normalisierung (im Wesentlichen durch 1024/Betrag im Quadrat der Basisfunktion) in dem im Folgenden beschriebenen Vorwärtstransformationsprozess durchgeführt wird.

[0095] Gemäß der oben geführten Diskussion zu der Bereichsbeschränkung hat der Transformationsbereichs-Datenblock D an dem Eingang der inversen Transformation in der illustrierten Transformations-Implementierung einen Bereich von 12 Bits. (Aufgrund der Normalisierung wird der Bereich des Transformationsbereichs-Datenblockes, der von der Vorwärtstransformation erzeugt wird, tatsächlich auf ± 2048/1,12 reduziert.) Die vorschriftsmäßige Zwischenmatrix (nach der inversen Transformation in einer Dimension) ist

$$D_1 = \frac{(D \cdot T_8)}{32}$$

und hat einen Bereich von etwas weniger als 10,5 Bits.

[0096] In einer inversen Transformations-Implementierung **500** können die Skalierungsstufen **520** und **540** jeweils die Datenblockwerte um 5 Bitstellen (effektiv eine Division durch 32) abrunden oder verschieben, wodurch insgesamt eine Verschiebung um 10 Bitstellen (für eine Division durch 1024) erreicht wird. Dadurch werden die Bereichsbeschränkungen in den Multiplikationsstufen **510** und **530** innerhalb eines 16-Bit-Bereichs aufrechterhalten.

[0097] Eine alternative Implementierung bewahrt arithmetische Präzision für die zweite, die Nachher-Multiplikationsstufe, indem die Menge des Skalierens und des Rundens in der ersten Skalierungsstufe **520** reduziert

wird. Da der Ergebnis-Datenblock D_1 nach der ersten Multiplikationsstufe einen Bereich von 10,5 Bits verwendet, und die 8-Punkt-Transformation den Bereich um 4 Bits erweitert, kann höchstens 1 Bit weniger des Skalierens in der ersten Skalierungsstufe zugelassen werden, um die 16-Bit-Bereichsbeschränkung in beiden Multiplikationsstufen einzuhalten. Demzufolge kann die alternative Implementierung eine Verschiebung um 4 Bits in der ersten Skalierungsstufe **520** und eine Verschiebung um 6 Bits in der zweiten Skalierungsstufe **540** bewirken. Die vorschriftsmäßige Darstellung der Transformation lautet nun:

$$D_1 = \frac{(D \cdot T_8)}{16}$$

$$R = \frac{(T_4^c \cdot D_1)}{64}$$

[0098] Diese alternative Implementierung lässt zu, dass ein zusätzlicher Präzisionsbit an D_1 einbehalten wird. Die gleiche Skalierung kann für die 4-Punkt-Transformation verwendet werden, da der größte Faktor in der 4-Punkt-Transformationsmatrix immer noch, obgleich sehr knapp, innerhalb des verfügbaren Headrooms (4,5 Bits) liegt.

[0099] Zum Einbehalten eines zweiten zusätzlichen Präzisionsbits an dem Zwischen-Datenblock D_1, zerlegt eine weitere alternative Implementierung die Transformationsmatrix wie folgt:

$T_8 = 2 \cdot T_8^0 + T_8^0$

$T_4 = 2 \cdot T_4^c + T_4^0$

[0100] Hierbei können die Matrizen der ungeraden Komponenten T_8^0 und T_4^0 lediglich die Werte 0, 1 und –1 als Einträge haben. Da die meisten der Einträge von T_8 gerade sind, ist T_8^0 eine Matrix mit leerem Bereich. Auf die gleiche Weise hat T_4^0 eine Struktur die in hohem Maße mit T_4^C korreliert. Die vorschriftsmäßige Darstellung dieser weiteren alternativen inversen Transformations-Implementierung wird nun wie folgt definiert:

$$D_1 = \frac{(D \cdot T_8)}{8}$$

$$R = \frac{\left(T_4^{ee} \cdot D_1 + \frac{T_8^{eo} \cdot D_1}{2}\right)}{64}$$

[0101] Die erste Skalierungsstufe **520** bewirkt nun eine Verschiebung nach unten um lediglich 3 Bits, wobei ein extra Präzisionsbit für die zweite Multiplikationsstufe **530** einbehalten wird. Da die gerade Komponente die Hälfte des Bereiches von T_8 hat, und da die ungerade Komponente T_8^0 darauf beschränkt ist, 0, 1 und –1 als Einträge zu haben, besitzt der resultierende Zähler eine Beschränkung des Bereichs auf 16 Bits. Es gibt eine geringfügige Rechenstrafe, die für den zusätzlichen Präzisionsbit an D_1 zu entrichten ist. Nichtsdestotrotz resultiert diese Zerlegung der Transformationsmatrix in einer verbesserten arithmetischen Genauigkeit zu verschwindend geringen Kosten.

[0102] In anderen alternativen Implementierungen kann die Transformation T im allgemeineren Sinne in Komponenten-Transformationsmatrizen T_a und T_b zerlegt werden, die in Bezug zu der Transformationsbasis T als $T = 2^x \times T_a + T_b$ stehen. Die Gleichung der Transformation

$$D_1 = \frac{D \cdot T}{2^y},$$

kann dann definiert sein als:

$$R = \frac{\left(D \cdot T_a + \frac{D \cdot T_b}{2^x}\right)}{2^{y-x}}$$

[0103] Diese Gleichung kann unter Verwendung der Matrixmultiplikations- und Verschiebungsoperation wie folgt implementiert werden: $D \cdot T_a + ((D \cdot T_b) >> x)) >> (y - x)$.

[0104] In nochmaligem Bezug auf die bestimmte alternative Implementierung der Transformation T_8 werden

die ungeraden und die geraden Komponenten der 8-Punkt-Transformation wie untenstehend dargestellt:

$$T_8^e = \begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\ 8 & 7 & 4 & 2 & -2 & -4 & -7 & -8 \\ 8 & 3 & -3 & -8 & -8 & -3 & 3 & 8 \\ 7 & -2 & -8 & -5 & 5 & 8 & 2 & -7 \\ 6 & -6 & -6 & 6 & 6 & -6 & -6 & 6 \\ 4 & -8 & 2 & 7 & -7 & -2 & 8 & -4 \\ 3 & -8 & 8 & -3 & -3 & 8 & -8 & 3 \\ 2 & -5 & 7 & -8 & 8 & -7 & 5 & -2 \end{bmatrix}$$

$$T_8^o = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & -1 & -1 & 0 \end{bmatrix}$$

[0105] Hierbei ist zu beachten, dass T_8^0 lediglich zwei eigenständige nicht-Null-Spalten hat. Nachher-Multiplikation durch T_8^0 ist gleichbedeutend mit kaum zwei Additionen (und Negationen):

$W \times T_8^0 = [W_1\ W_2\ W_2\ W_1 - W_1 - W_2 - W_2 - W_1]$

wobei

$$[W_0\ \ W_2] = W \cdot \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

[0106] Die geraden und die ungeraden Komponenten der inversen 4-Punkt-Transformation sind:

$$T_4^e = \begin{bmatrix} 8 & 8 & 8 & 8 \\ 11 & 5 & -5 & -11 \\ 8 & -8 & -8 & 8 \\ 5 & -11 & 11 & -5 \end{bmatrix}$$

$$T_4^o = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

[0107] In dieser weiteren alternativen Implementierung führt die zeilenweise oder die Vorher-Multiplikationsstufe **510** und die erste Skalierungsstufe **520** die folgende Operation für die inverse 8 × 8-Transformation durch:

$D_1 = (D \times T_8 + 4) >> 3$

[0108] Die spaltenweise oder die Nachher-Multiplikationsstufe **530** und die Skalierungsstufe **540** in dieser inversen Transformations-Implementierung ist zunächst durch Berücksichtigung der ungeraden Komponente von T_8 zum Berechnen der zwei gemeinsamen Zeilen von 8 Elementen definiert. Diese werden um ein Bit nach

rechts verschoben und anschließend zu dem geraden Komponentenprodukt hinzugefügt (oder von ihm subtrahiert), bevor das Ergebnis um 6 Bits nach unten abgerundet wird. Die Operationen, die in diesen Stufen durchgeführt werden, werden dementsprechend wie folgt ausgedrückt:

$$[D_{1a} \quad D_{1b}] = D_1' \cdot \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$D_{2a} = D_{1a} >> 1$$

$$D_{2b} = D_{1b} >> 1$$

$$R = \left(T_8^{tr} \cdot D_1 + \begin{bmatrix} D_{2a} \\ D_{2b} \\ D_{2b} \\ D_{2a} \\ -D_{2a} \\ -D_{2b} \\ -D_{2b} \\ -D_{2a} \end{bmatrix} + 32 \right) >> 6$$

[0109] Für die inverse 4 × 8-Transformation in der weiteren alternativen Implementierung (die sich entsprechend der in diesem Dokument verwendeten Konvention auf eine Gruppe mit 4 Spalten und 8 Zeilen bezieht) führen die zeilenweise oder die Vorher-Multiplikationsstufe **510** und die Skalierungsstufe **520** eine 4-Punkt-Operation durch, die folgendermaßen definiert wird:

$D_1 = (D \times T_4 + 4) >> 3$

[0110] Für den zweiten Teil der Transformation ist die spaltenweise oder die Nachher-Multiplikationsstufe **530** und die Skalierungsstufe **540** identisch mit der für die inverse 8 × 8-Transformation, die soeben beschrieben worden ist.

[0111] Für die inverse 8 × 4-Transformation arbeitet die weitere alternative Implementierung in der zeilenweisen Stufe **510** und in der Skalierungsstufe **520** an den 4 Zeilen/8 Spalten der Transformationsbereichs-Daten entsprechend der folgenden Gleichung:

$D_1 = (D \times T_8 + 4) >> 3$

[0112] Die inverse 4-Punkt-Transformation für die spaltenweise Stufe **530** und die Skalierungsstufe **540** wird wie untenstehend definiert:

$$[D_{1a} \quad D_{1b}] = D_1' \cdot \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 1 & -1 \\ 0 & 0 \end{bmatrix}$$

$$D_{2a} = D_{1a} >> 1$$
$$D_{2b} = D_{1b} >> 1$$

$$R = \left(T_4'' \cdot D_1 + \begin{bmatrix} D_{2a} \\ D_{2b} \\ -D_{2b} \\ -D_{2a} \end{bmatrix} + 32 \right) >> 6$$

[0113] Für die inverse 4 × 4-Transformation werden die Stufen **510**, **530** der inversen 4 × 4-Transformation wie oben beschrieben jeweils für die zeilenweise Stufe **510** der inversen 4 × 8-Transformation und für die spaltenweise Stufe **530** der inversen 8 × 4-Transformation durchgeführt.

Implementierung der Vorwärtstransformation

[0114] Bei der Vorwärtstransformation können auch Implementierungen mit Bereichsbeschränkung, so wie die in Fig. 5 dargestellten, einschließlich Skalierungsstufen angewendet werden, in dem Kontext der Videokodierung ist eine solche Vorgehensweise jedoch oftmals redundant. Da der Kodierungsprozess langsamer von statten geht als das Dekodieren und die verfügbarem Rechenressourcen im Kodierer oftmals eine Größenordnung höher als die in dem Dekodierer sind, kann der Kodierer in Integern höheren Präzision (beispielsweise 32 Bit) oder sogar in Gleitkommas einer zweifachen Präzision implementiert sein. Mit anderen Worten bedeutet dies, dass die Skalierungsstufen **520**, **540** in der Implementierung der Vorwärtstransformation weggelassen werden können.

[0115] In einer in Fig. 8 dargestellten bevorzugten Implementierung der Vorwärtstransformation wird eine Stufe der erneuten Normalisierung **740** auf der Vorwärtstransformations-Seite durchgeführt (nach der zeilenweisen und der spaltenweisen Multiplikation **710** und **730**), um sicherzustellen, dass die Transformationen selbst eine minimale Fehlermenge bei der Rekonstruierung verursachen. Dadurch wird die leichte Abweichung bei den Beträgen der Basisfunktionen kompensiert und der Rechenaufwand in dem Dekodierer reduziert, in dem möglicherweise weniger Rechenressourcen verfügbar sind. Alternativ dazu kann für Systeme, die über ausreichende Rechenressourcen in dem Dekodierer verfügen die Kompensation (Skalierung der Transformationsdaten) in dem Dekodierer durchgeführt werden.

[0116] In der folgenden Diskussion bezeichnet das Symbol ⊗ das komponentenweise Produkt der Matrizen gleicher Größe. Das ursprüngliche 2-D-Signal (Sample-Datenblock) wird mit X bezeichnet, und seine Transformation wird mit Y bezeichnet (X_1 bezeichnet den Zwischenblock nach der zeilenweisen Transformation). Die Matrizen X und Y haben dieselbe Größe und decken die Fälle der 8 × 8-, 8 × 4-, und der 4 × 8-Vorwärtstransformation ab. Es werden die folgenden Skalierungsfaktoren verwendet:

$$f_0 = \frac{64}{288 \times 288}$$

$$f_1 = \frac{64}{288 \times 289}$$

$$f_2 = \frac{64}{289 \times 289}$$

$$f_3 = \frac{64}{288 \times 292}$$

$$f_4 = \frac{64}{289 \times 292}$$

$$f_5 = \frac{64}{292 \times 292}$$

[0117] Für die 8 × 8-Vorwärtstransformation wird die Verarbeitung in den Stufen **710**, **730** und **740** wie folgt beschrieben:

$$Y = (T_8 \cdot X \cdot T_8') \otimes \begin{bmatrix} f_0 & f_1 & f_3 & f_1 & f_0 & f_1 & f_3 & f_1 \\ f_1 & f_2 & f_4 & f_2 & f_1 & f_2 & f_4 & f_2 \\ f_3 & f_4 & f_5 & f_4 & f_3 & f_4 & f_5 & f_4 \\ f_1 & f_2 & f_4 & f_2 & f_1 & f_2 & f_4 & f_2 \\ f_0 & f_1 & f_3 & f_1 & f_0 & f_1 & f_3 & f_1 \\ f_1 & f_2 & f_4 & f_2 & f_1 & f_2 & f_4 & f_2 \\ f_3 & f_4 & f_5 & f_4 & f_3 & f_4 & f_5 & f_4 \\ f_1 & f_2 & f_4 & f_2 & f_1 & f_2 & f_4 & f_2 \end{bmatrix}$$

[0118] Für die 4 × 8-Vorwärtstransformation, führen die Stufen folgenden Operationen durch:

$$Y = (T_8 \cdot X \cdot T_4') \otimes \begin{bmatrix} f_1 & f_3 & f_1 & f_3 \\ f_2 & f_4 & f_2 & f_4 \\ f_4 & f_5 & f_4 & f_5 \\ f_2 & f_4 & f_2 & f_4 \\ f_1 & f_3 & f_1 & f_3 \\ f_2 & f_4 & f_2 & f_4 \\ f_4 & f_5 & f_4 & f_5 \\ f_2 & f_4 & f_2 & f_4 \end{bmatrix}$$

[0119] Die 8 × 4-Vorwärtstransformation ist andererseits auch die Umstellung der oben beschriebenen 4 × 8-Vorwärtstransformation.

[0120] Die 4 × 4-Vorwärtstransformation führt die Skalierung wie folgt durch:

$$Y = (T_4 \cdot X \cdot T_4') \otimes \begin{bmatrix} f_2 & f_4 & f_2 & f_4 \\ f_4 & f_5 & f_4 & f_5 \\ f_2 & f_4 & f_2 & f_4 \\ f_4 & f_5 & f_4 & f_5 \end{bmatrix}$$

VI. Geeignete Rechenumgebung

[0121] Die oben beschriebenen Transformationen können in einem beliebigen von einer Auswahl von Geräten implementiert werden, in denen Bild- und Videosignalverarbeitung durchgeführt wird, einschließlich Computern; Bild- und Videoaufzeichnungsgeräten, Sende- und Empfangsgeräten; tragbaren Videoabspielgeräten; Videokonferenzgeräten; und so weiter. Die Verfahren zum Kodieren der Bild- und Videodaten können in Hardwareschaltungen aber ebenso auch in Bild- und Videoverarbeitungssoftware implementiert werden, die innerhalb einer Computer- oder einer anderen Rechenumgebung, wie in Fig. 8 dargestellt, arbeiten.

[0122] Fig. 8 illustriert ein allgemeines Beispiel einer geeigneten Rechenumgebung (**800**), in der die beschriebenen Ausführungsbeispiele implementiert werden können. Es ist nicht die Absicht, dass die Rechenumgebung (**800**) irgendeine Beschränkung hinsichtlich des Nutzungsumfanges oder der Funktionalität der Erfindung vorschlägt, da die vorliegende Erfindung in verschiedenen allgemeinen oder speziellen Rechenumgebungen implementiert werden kann.

[0123] In Bezug auf Fig. 8 umfasst die Rechenumgebung (**800**) wenigstens eine Verarbeitungseinheit (**810**) und einen Speicher (**820**). In Fig. 8 ist diese grundlegendste Konfiguration (**830**) innerhalb einer gestrichelten Linie dargestellt. Die Verarbeitungseinheit (**810**) führt Computer-ausführbare Instruktionen aus und kann ein realer oder ein virtueller Prozessor sein. In einem Simultanverarbeitungssystem führen simultane Verarbeitungseinheiten Computer-ausführbare Instruktionen aus, um die Verarbeitungsleistung zu erhöhen. Der Speicher (**820**) kann ein flüchtiger Speicher (beispielsweise Register, Cache-Speicher, Schreib-/Lesespeeicher RAM), ein nicht-flüchtiger Speicher (beispielsweise Festwertspeicher ROM, elektrisch löschbarer programmierbarer Lesespeicher EEPROM, Flash-Speicher und so weiter) oder eine Kombination aus beiden sein. Der Speicher (**820**) speichert die Software (**880**), die die beschriebenen Bild- und/oder Videokodierer/Dekodierer und die Transformationen implementiert.

[0124] Eine Rechenumgebung kann darüber hinaus weitere Leistungsmerkmale umfassen. So enthält die Rechenumgebung (**800**) beispielsweise einen Speicher (**840**), ein oder mehrere Eingabegeräte (**850**), ein oder mehrere Ausgabegeräte (**860**) und eine oder mehrere Kommunikationsverbindungen (**870**). Eine Vorrichtung zur Zusammenschaltung (nicht dargestellt) wie beispielsweise ein Bus, ein Kontroller oder ein Netzwerk schaltet die Komponenten der Rechenumgebung (**800**) zusammen. Typischerweise stellt die Betriebssystemsoftware (nicht dargestellt) eine Betriebsumgebung für weitere Software bereit, die in der Rechenumgebung (**800**) arbeitet und die Aktivitäten der Komponenten der Rechenumgebung (**800**) koordiniert.

[0125] Der Speicher (**840**) kann ein entnehmbarer oder ein nicht-entnehmbarer Speicher sein und umfasst Magnetplatten, Magnetbänder- oder kassetten, CD-ROMs, CD-RWs, DVDs oder ein beliebiges anderes Medium, das zum Speichern von Informationen verwendet werden kann, auf die innerhalb der Rechenumgebung zugegriffen werden kann. Der Speicher (**840**) speichert die Instruktionen für die Software (**880**), die den Audiokodierer implementiert, der die Quantisierungsmatrizen erzeugt und komprimiert.

[0126] Das/die Eingabegeräte (**850**) kann ein Touch-Eingabegerät wie beispielsweise eine Tastatur, eine Maus, ein Pen oder Trackball, ein Spracheingabegerät, ein Scanngerät oder eine andere Vorrichtung sein, mit der eine Eingabe in die Rechenumgebung (**800**) getätigt werden kann. Für Audio kann das/die Eingabegerät/e eine Soundkarte oder eine ähnliche Vorrichtung, die eine Audioeingabe in analoger oder digitaler Form akzeptiert, oder ein CD-ROM-Leser sein, der der Rechenumgebung Audiosamples zur Verfügung stellt. Das/die Ausgabegerät/e (**860**) kann ein Anzeige, ein Drucker, ein Lautsprecher, eine CD-Schreibvorrichtung oder ein anderes Gerät sein, das eine Ausgabe von der Rechenumgebung (**800**) ermöglicht.

[0127] Die Kommunikationsverbindung(en) (**870**) ermöglicht die Kommunikation über ein Kommunikationsmedium zu einer anderen Recheneinheit. Das Kommunikationsmedium überträgt Informationen wie beispielsweise Computer-ausführbare Instruktionen, komprimierte Audio- oder Videoinformationen oder andere Daten in einem modulierten Datensignal. Ein moduliertes Datensignal ist ein Signal, das ein oder mehrere seiner Eigenschaften so eingestellt oder geändert hat, dass es die in dem Signal enthaltenen Informationen kodiert. Um ein Beispiel zu nennen, was allerdings nicht im restriktiven Sinne zu betrachten ist, umfassen Kommunikationsmedien kabelbasierte oder kabellose Verfahren, die mit einem elektrischen, einem optischen, einem RF-, einem Infrarot-, einem akustischen oder einem anderen Träger implementiert sind.

[0128] Die in diesem Dokument beschriebenen Transformations- und Kodierungs-/Dekodierungsverfahren können in dem allgemeinen Kontext von Computer-lesbaren Medien beschrieben werden. Computer-lesbare Medien sind jede beliebigen verfügbaren Medien, auf die innerhalb einer Rechenumgebung zugegriffen wer-

den kann. Um ein Beispiel zu nennen, was allerdings nicht im restriktiven Sinne zu betrachten ist, umfassen mit der Rechenumgebung (**800**) Computer-lesbare Medien Speicher (**820**), Speichervorrichtungen (**840**), Kommunikationsmedien und Kombinationen aus beliebigen der oben genannten.

[0129] Die in diesem Dokument beschriebenen Transforrations- und Kodierungs-/Dekodierungsverfahren können in dem allgemeinen Kontext von Computerausführbaren Instruktionen, wie beispielsweise die, die in den Programmmodulen enthalten sind, die in einer Rechenumgebung auf einem realen oder virtuellen Zielprozessor ausgeführt werden, beschrieben werden. Im Allgemeinen umfassen Programmmodule Routinen, Programme, Bibliotheken, Objekte, Klassen, Komponenten, Datenstrukturen und so weiter, die bestimmte Aufgaben ausführen oder bestimmte abstrakte Datentypen implementieren. Die Funktionen der Programmmodule können zwischen den Programmmodulen kombiniert oder aufgeteilt werden, so wie die in jeweiligen Ausführungsbeispielen gewünscht wird. Computer-ausführbare Instruktionen für Programmmodule können innerhalb einer lokalen oder verteilten Rechenumgebung ausgeführt werden.

[0130] Im Sinne der Darstellung verwendet die ausführliche Beschreibung Begriffe wie „bestimmen", „erzeugen", „anpassen" und „anwenden", um in einer Rechenumgebung ausgeführte Computeroperationen zu beschreiben. Bei diesen Begriffen handelt es sich um Abstraktionen auf hoher Ebene, und sie beziehen sich auf Operationen, die durch einen Computer ausgeführt werden, weshalb sie nicht mit Handlungen, die durch eine Person ausgeführt werden, verwechselt werden sollten. Die eigentlichen Operationen, die diesen Begriffen entsprechen, variieren je nach Implementierung.

VII. Erweiterungen

[0131] Die oben beschriebenen Transformationen und ihre Implementierungen können auf verschiedene Art und Weise erweitert werden, dazu gehören die folgenden Möglichkeiten:
Abwandlungen der Transformations-Implementierungen können abwechselndes Runden, „Butterfly"-Implementierung und biorthogonale Formeln umfassen.

[0132] Darüber hinaus kann der oben beschriebene Konstruierungsprozess so weitergeführt werden, das er unterschiedlichen Bitbereichsbeschränkungen unterliegt, um Transformationsvarianten mit unterschiedlichen Koeffizientenbereichen zu erzeugen.

[0133] Die Schwellenwertparameter der Beschränkungen, die in dem Konstruierungsprozess verwendet werden, können so variiert werden, dass sie einen größeren Satz von Beanstandungstransformationen enthalten.

[0134] Der Transformations-Satz kann so erweitert werden, dass er weitere Blockgrößen enthält (beispielsweise die 4 × 4-Block- oder Transformationen anderer Blockgrößen). Auf die gleiche Weise muss der erzeugte Transformationssatz nicht notwendigerweise die Größen der oben illustrierten Implementierungen (das heißt die 8 × 8-, die 8 × 4- und die 4 × 8-Transformation) enthalten. Die Transformationen können auch auf andere Blockgrößen und höhere Dimensionen (> 2) ausgedehnt werden.

[0135] Die Wahl der Bittiefe ist ein variabler Parameter und kann beliebig für das illustrierte Beispiel geändert werden (mit einer 8-Bit-Aiflösung von unverarbeiteten Pixelwerten und 16-Bits für Zwischenergebnisse).

[0136] Die Implementierung der Vorwärtstransformation kann ebenfalls einer Bereichsbeschränkung unterliegen, indem die Vorgehensweise angewendet wird, die für die Bereichsbeschränkung der Implementierung der inversen Transformation beschrieben wurde.

[0137] Der oben beschriebene Konstruierungsprozess kann so weitergeführt werden, dass er Transformationen erzeugt, die annähernd anderen jedoch Nicht-DCT-Basisfunktionen entsprechen. So kann der Konstruierungsprozess beispielsweise Transformationen erzeugen, die annähernd FFT-, Wavelet-, oder anderen überlappenden Transformationen entsprechen, wobei die Beschränkungen zu skalierten Integerzahlen, der perfekten Rekonstruierung und die Bereichsbeschränkung erfüllt werden.

[0138] In Anbetracht der beschriebenen und illustrierten Prinzipien unserer Erfindung in Bezug auf die beschriebenen Ausführungsbeispiele wird ersichtlich, dass die beschriebenen Ausführungsbeispiele im Aufbau und Detail modifiziert werden können, ohne dabei von solchen Prinzipien abzuweichen. Es wird darauf hingewiesen, dass die in diesem Dokument beschriebenen Programme, Prozesse oder Verfahren nicht in Beziehung zu einem bestimmten Typ von Rechenumgebung stehen oder auf diesen beschränkt sind, es sei denn es wird anderweitig darauf hingewiesen. Es können verschiedene Typen von allgemeinen oder speziellen Re-

chenumgebungen für die Ausführung der Operationen in Übereinstimmung mit den in diesem Dokument beschriebenen Schritten verwendet werden. Es können Elemente der beschriebenen Ausführungsbeispiele, die in der Software dargestellt sind, auch in der Hardware implementiert sein und umgekehrt.

[0139] Während die Transformations-Kodierverfahren in diesem Dokument an Stellen als Bestandteil eines einzigen, integrierten Systems beschrieben sind, können die Verfahren auch separat voneinander und möglicherweise in Kombination mit anderen Verfahren angewendet werden. So implementiert in alternativen Ausführungsbeispielen ein Signalverarbeitungswerkzeug mit Ausnahme von einem Kodierer oder einem Dekodierer eins oder mehrere der Verfahren.

[0140] Die beschriebenen Transformationen und Transformations-Kodierungs-Ausführungsbeispiele führen verschiedene Verfahren durch. Obgleich die Operationen für diese Verfahren typischerweise im Sinne der Darstellung in einer bestimmten sequenziellen Reihenfolge beschrieben sind, sollte hierbei angemerkt werden, dass diese Art und Weise der Beschreibung auch geringfügige Umstellungen in der Reihenfolge der Operationen umfasst, es sei denn, es wird eine bestimmte Reihenfolge verlangt. So können beispielsweise Operationen, die in einer bestimmten Reihenfolge beschrieben sind, in einigen Fällen in eine andere Reihenfolge gebracht werden oder auch gleichzeitig durchgeführt werden. Darüber hinaus zeigen der Einfachheit halber Ablaufpläne typischerweise nicht die verschiedenen Möglichkeiten, in denen die bestimmten Verfahren zusammen mit anderen Verfahren verwendet werden können.

[0141] In Anbetracht der vielen möglichen Ausführungsbeispiele, auf die die Prinzipien unserer Erfindung angewendet werden können, erheben wir Anspruch auf alle solche Ausführungsbeispiele, die in den Umfang der folgenden Ansprüche fallen können.

Patentansprüche

1. Verfahren zum Umwandeln eines Mediendatenblocks, der die Dimensionen n und m hat, umfassend:
Erstellen einer Transformationsmatrix, die aus Integerkoeffizienten besteht, die eine zweite Transformationsmatrix, die Nicht-Integerkoeffizienten hat, gemäß vorbestimmter Toleranzen und vorbestimmter Beschränkungen annähert; und
Ausführen einer Transformation durch das Ausführen mindestens einer Matrixmultiplikation des Mediendatenblocks mit der Transformationsmatrix, die aus Integerkoeffizienten besteht, zum Erlangen eines transformierten Mediendatenblocks;
dadurch gekennzeichnet, dass der Schritt des Erstellens einer Transformationsmatrix die folgenden Schritte umfasst:
(a) Ermitteln eines konstanten Faktors aus n- und m-Punkt-DC-Basisfunktionen, die innerhalb einer Toleranz Beschränkungen unterliegen, dass Beträge der n- und m-Punkt-DC-Basisfunktionen übereinstimmen, und dass Basisfunktionen Transformationsbereichsdaten innerhalb eines begrenzten Bereichs von Integerwerten erzeugen;
(b) Ermitteln einer Reihe von konstanten Faktoren aus ungeraden Basisfunktionen von einem grösseren der n- oder m-Punkte, die innerhalb einer Toleranz Beschränkungen unterliegen, dass solche ungeraden Basisfunktionen orthogonal sind, zu entsprechenden konstanten Basisfunktionsfaktoren der zweiten Transformationsmatrix gut korrelieren und mit den DC-Basisfunktionen betragsmässig übereinstimmen; und
(c) Ermitteln einer Reihe von konstanten Faktoren aus geraden Basisfunktionen von dem grösseren der n- oder m-Punkte, die innerhalb einer Toleranz Beschränkungen unterliegen, dass solche geraden Basisfunktionen orthogonal sind, zu entsprechenden konstanten Basisfunktionsfaktoren der zweiten Transformationsmatrix gut korrelieren und mit den DC- und ungeraden Basisfunktionen betragsmässig übereinstimmen.

2. Verfahren gemäß Anspruch 1, das des Weiteren die folgenden Schritte umfasst:
Quantisieren des transformierten Mediendatenblocks;
Dequantisieren der transformierten Mediendaten; und
Ausführen einer Invers-Transformation, die als eine Sequenz von Matrixmultiplikationen implementiert ist.

3. Verfahren gemäß Anspruch 2, wobei die Beschränkungen auch eine Beschränkung, dass Basisfunktionen der Transformation betragsmässig nahe beieinander sind, und eine Beschränkung, dass es genügend Headroom gibt, umfassen.

4. Verfahren gemäß Anspruch 3, das des Weiteren das Ausgleichen irgendwelcher Ungleichheiten der Basisfunktionsbeträge bei der Quantisierung umfasst.

5. Verfahren gemäß Anspruch 3, das des Weiteren das Ausgleichen irgendwelcher Ungleichheiten der Basisfunktionsbeträge bei der Dequantisierung umfasst.

6. Verfahren gemäß Anspruch 3, das des Weiteren das Ausgleichen irgendwelcher Ungleichheiten der Basisfunktionsbeträge zum Teil bei der Quantisierung und zum Teil bei der Dequantisierung umfasst.

7. Verfahren gemäß Anspruch 1, wobei mindestens eine Dimension des Mediendatenblocks 8 Punkte hat, der Schritt des Erstellens zu einer 8-Punkt-Transformationsmatrix

$$T_8 = \begin{bmatrix} 12 & 12 & 12 & 12 & 12 & 12 & 12 & 12 \\ 16 & 15 & 9 & 4 & -4 & -9 & -15 & -16 \\ 16 & 6 & -6 & -16 & -16 & -6 & 6 & 16 \\ 15 & -4 & -16 & -9 & 9 & 16 & 4 & -15 \\ 12 & -12 & -12 & 12 & 12 & -12 & -12 & 12 \\ 9 & -16 & 4 & 15 & -15 & -4 & 16 & -9 \\ 6 & -16 & 16 & -6 & -6 & 16 & -16 & 6 \\ 4 & -9 & 15 & -16 & 16 & -15 & 9 & -4 \end{bmatrix}$$

führt und das Verfahren des Weiteren den Schritt des Skalierens des resultierenden Matrixprodukts umfasst, um innerhalb eines Bit-Bereichslimits zu bleiben.

8. Verfahren gemäß Anspruch 7, wobei der Mediendatenblock ein 8 × 8-Block ist, und das Ausführen mindestens einer Matrixmultiplikation das Ausführen von reihenweisen und spaltenweisen Matrixmultiplikationen des Mediendatenblocks mit der Transformationsmatrix umfasst.

9. Verfahren gemäß Anspruch 8, wobei das Skalieren eine eintragsweise Verschiebungsfunktion nach jeder der reihenweisen und spaltenweisen Matrixmultiplikationen umfasst, um eine Teilung durch eine Zweierpotenz zu bewirken.

10. Verfahren gemäß Anspruch 9, wobei die eintragsweise Verschiebungsfunktion nach einer ersten der Matrixmultiplikationen eine Verschiebung durch weniger Bit-Positionen ist als nach einer zweiten der Matrixmultiplikationen.

11. Verfahren gemäß Anspruch 9, wobei die eintragsweisen Verschiebungsfunktionen Verschiebungen durch eine gleiche Anzahl von Bit-Positionen nach jeder der Matrixmultiplikationen sind.

12. Verfahren gemäß Anspruch 7, wobei der Schritt des Skalierens das Ausführen eines komponentenweisen Produkts unter Verwendung einer Matrix

$$\begin{bmatrix} f_0 & f_1 & f_3 & f_1 & f_0 & f_1 & f_3 & f_1 \\ f_1 & f_2 & f_4 & f_2 & f_1 & f_2 & f_4 & f_2 \\ f_3 & f_4 & f_5 & f_4 & f_3 & f_4 & f_5 & f_4 \\ f_1 & f_2 & f_4 & f_2 & f_1 & f_2 & f_4 & f_2 \\ f_0 & f_1 & f_3 & f_1 & f_0 & f_1 & f_3 & f_1 \\ f_1 & f_2 & f_4 & f_2 & f_1 & f_2 & f_4 & f_2 \\ f_3 & f_4 & f_5 & f_4 & f_3 & f_4 & f_5 & f_4 \\ f_1 & f_2 & f_4 & f_2 & f_1 & f_2 & f_4 & f_2 \end{bmatrix}$$

ist, wobei gilt:

$$f_0 = \frac{64}{288 \times 288}$$

$$f_1 = \frac{64}{288 \times 289}$$

$$f_2 = \frac{64}{289 \times 289}$$

$$f_3 = \frac{64}{288 \times 292}$$

$$f_4 = \frac{64}{289 \times 292}$$

$$f_5 = \frac{64}{292 \times 292}.$$

13. Verfahren gemäß Anspruch 1, wobei mindestens eine Dimension des Mediendatenblocks 4 Punkte hat, der Schritt des Erstellens zu einer 4-Punkt-Transformationsmatrix

$$T_4 = \begin{bmatrix} 17 & 17 & 17 & 17 \\ 22 & 10 & -10 & -22 \\ 17 & -17 & -17 & 17 \\ 10 & -22 & 22 & -10 \end{bmatrix}$$

führt und das Verfahren des Weiteren den Schritt des Skalierens des resultierenden Matrixprodukts umfasst, um innerhalb eines Bit-Bereichslimits zu bleiben.

14. Verfahren gemäß Anspruch 13, wobei der Mediendatenblock ein 4 × 4-Block ist, und das Ausführen mindestens einer Matrixmultiplikation das Ausführen von reihenweisen und spaltenweisen Matrixmultiplikationen des Mediendatenblocks mit der Transformationsmatrix umfasst.

15. Verfahren gemäß Anspruch 14, wobei das Skalieren eine eintragsweise Verschiebungsfunktion nach jeder der reihenweisen und spaltenweisen Matrixmultiplikationen umfasst, um eine Teilung durch eine Zweierpotenz zu bewirken.

16. Verfahren gemäß Anspruch 14, wobei die eintragsweise Verschiebungsfunktion nach einer ersten der Matrixmultiplikationen eine Verschiebung durch weniger Bit-Positionen ist als nach einer zweiten der Matrixmultiplikationen.

17. Verfahren gemäß Anspruch 14, wobei die eintragsweisen Verschiebungsfunktionen Verschiebungen durch eine gleiche Anzahl von Bit-Positionen nach jeder der Matrixmultiplikationen sind.

18. Verfahren gemäß Anspruch 13, wobei der Schritt des Skalierens das Ausführen eines komponentenweisen Produkts unter Verwendung einer Matrix

$$\begin{bmatrix} f_2 & f_4 & f_2 & f_4 \\ f_4 & f_5 & f_4 & f_5 \\ f_2 & f_4 & f_2 & f_4 \\ f_4 & f_5 & f_4 & f_5 \end{bmatrix}$$

ist, wobei gilt:

$$f_2 = \frac{64}{289 \times 289}$$

$$f_4 = \frac{64}{289 \times 292}$$

$$f_5 = \frac{64}{292 \times 292}.$$

19. Verfahren gemäß Anspruch 1, das des Weiteren den folgenden Schritt umfasst:
(d) Ermitteln einer Reihe von Faktoren aus Basisfunktionen von einem kleineren der n- oder m-Punkte, die auch innerhalb der Toleranz Beschränkungen unterliegen, dass solche Basisfunktionen orthogonal sind, zu entsprechenden konstanten Basisfunktionsfaktoren der zweiten Transformationsmatrix gut korrelieren und mit den gösseren der n- oder m-Punkte-Basisfunktionen betragsmässig übereinstimmen.

20. Verfahren gemäß Anspruch 19, wobei die zweite Transformation eine DCT-Transformation ist.

21. Verfahren gemäß Anspruch 19, wobei die zweite Transformation eine DCT-, FFT-, Wavelet- oder Overlapped-Transformation ist.

22. Verfahren gemäß Anspruch 19, wobei die Dimensionen n und m gleich sind.

23. Verfahren gemäß Anspruch 19, wobei die Dimensionen n und m gleich 4 und 8 sind.

24. Verfahren gemäß Anspruch 19, wobei die Dimensionen des Mediendatenblocks 4 und 8 Punkte sind, und die Schritte (a) bis (c) zu einer 8-Punkt-Matrix der Form

$$T_8 = \begin{bmatrix} 12 & 12 & 12 & 12 & 12 & 12 & 12 & 12 \\ 16 & 15 & 9 & 4 & -4 & -9 & -15 & -16 \\ 16 & 6 & -6 & -16 & -16 & -6 & 6 & 16 \\ 15 & -4 & -16 & -9 & 9 & 16 & 4 & -15 \\ 12 & -12 & -12 & 12 & 12 & -12 & -12 & 12 \\ 9 & -16 & 4 & 15 & -15 & -4 & 16 & -9 \\ 6 & -16 & 16 & -6 & -6 & 16 & -16 & 6 \\ 4 & -9 & 15 & -16 & 16 & -15 & 9 & -4 \end{bmatrix}$$

führen, und der Schritt (d) zu einer 4-Punkt-Matrix der Form

$$T_4 = \begin{bmatrix} 17 & 17 & 17 & 17 \\ 22 & 10 & -10 & -22 \\ 17 & -17 & -17 & 17 \\ 10 & -22 & 22 & -10 \end{bmatrix}$$

führt, und der Schritt des Ausführens einer Transformation das Ausführen von reihenweisen und spaltenweisen Matrixmultiplikationen des Mediendatenblocks mit beiden Matrizen T_8 und T_4 umfasst.

25. Verfahren gemäß Anspruch 24, das des Weiteren den Schritt des Skalierens des resultierenden Matrixprodukts umfasst, um innerhalb eines Bit-Bereichslimits zu bleiben.

26. Verfahren gemäß Anspruch 24 oder 25, wobei der Datenblock Dimensionen von 4 × 8 Punkten hat, und der Vorgang des Ausführens der Matrixmultiplikationen gemäß der Relation $Y = (T_8 \cdot X \cdot T'_4)$ ausgeführt wird, wobei X den Datenblock darstellt und Y das resultierende Matrixprodukt ist.

27. Verfahren gemäß Anspruch 24 oder 25, wobei der Datenblock Dimensionen von 8 × 4 Punkten hat, und der Vorgang des Ausführens der Matrixmultiplikationen gemäß der Relation $Y = (T_4 \cdot X \cdot T'_8)$ ausgeführt wird, wobei X den Datenblock darstellt und Y das resultierende Matrixprodukt ist.

28. Verfahren gemäß Anspruch 25, wobei der Schritt des Skalierens das Ausführen eines komponentenweisen Produkts unter Verwendung einer Matrix

$$\begin{bmatrix} f_1 & f_3 & f_1 & f_3 \\ f_2 & f_4 & f_2 & f_4 \\ f_4 & f_5 & f_4 & f_5 \\ f_2 & f_4 & f_2 & f_4 \\ f_1 & f_3 & f_1 & f_3 \\ f_2 & f_4 & f_2 & f_4 \\ f_4 & f_5 & f_4 & f_5 \\ f_2 & f_4 & f_2 & f_4 \end{bmatrix}$$

ist, wobei gilt:

$$f_0 = \frac{64}{288 \times 288}$$

$$f_1 = \frac{64}{288 \times 289}$$

$$f_2 = \frac{64}{289 \times 289}$$

$$f_3 = \frac{64}{288 \times 292}$$

$$f_4 = \frac{64}{289 \times 292}$$

$$f_5 = \frac{64}{292 \times 292}.$$

29. Verfahren gemäß Anspruch 1, wobei die Transformationsmatrix T in $T = 2^x \cdot T_a + T_b$ zerlegt wird, wobei T_a und T_b Subkomponentenmatrizen sind, und wobei der Schritt des Ausführens der Transformation umfasst:
Ausführen von Matrixmultiplikationen des Datenblocks mit jeder von ersten und zweiten Transformations-Subkomponentenmatrizen;
Verschieben eines Produkts des Datenblocks und der zweiten Subkomponentenmatrix durch x Bit-Positionen;
Summieren eines Produkts des Datenblocks und der ersten Subkomponentenmatrix mit dem verschobenen Produkt des Datenblocks und der zweiten Subkomponentenmatrix; und
Verschieben einer Summe der Produkte durch y Bit-Positionen, um das Ergebnis

$$R = \frac{D \cdot T}{2^y}$$

mit D als dem Datenblock zu erzielen;
wodurch der Headroom der Transformation erweitert wird.

30. Verfahren gemäß Anspruch 29, wobei die Transformations-Basisfunktionsmatrix

$$T_8 = \begin{bmatrix} 12 & 12 & 12 & 12 & 12 & 12 & 12 & 12 \\ 16 & 15 & 9 & 4 & -4 & -9 & -15 & -16 \\ 16 & 6 & -6 & -16 & -16 & -6 & 6 & 16 \\ 15 & -4 & -16 & -9 & 9 & 16 & 4 & -15 \\ 12 & -12 & -12 & 12 & 12 & -12 & -12 & 12 \\ 9 & -16 & 4 & 15 & -15 & -4 & 16 & -9 \\ 6 & -16 & 16 & -6 & -6 & 16 & -16 & 6 \\ 4 & -9 & 15 & -16 & 16 & -15 & 9 & -4 \end{bmatrix}$$

ist; und
wobei die Transformations-Subkomponentenmatrizen

$$T_8^e = \begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\ 8 & 7 & 4 & 2 & -2 & -4 & -7 & -8 \\ 8 & 3 & -3 & -8 & -8 & -3 & 3 & 8 \\ 7 & -2 & -8 & -5 & 5 & 8 & 2 & -7 \\ 6 & -6 & -6 & 6 & 6 & -6 & -6 & 6 \\ 4 & -8 & 2 & 7 & -7 & -2 & 8 & -4 \\ 3 & -8 & 8 & -3 & -3 & 8 & -8 & 3 \\ 2 & -5 & 7 & -8 & 8 & -7 & 5 & -2 \end{bmatrix}$$

$$T_8^o = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & -1 & -1 & 0 \end{bmatrix}$$

sind.

31. Verfahren gemäß Anspruch 29, wobei die Transformations-Basisfunktionsmatrix

$$T_4 = \begin{bmatrix} 17 & 17 & 17 & 17 \\ 22 & 10 & -10 & -22 \\ 17 & -17 & -17 & 17 \\ 10 & -22 & 22 & -10 \end{bmatrix}$$

ist; und
wobei die Transformations-Subkomponentenmatrizen

$$T_4^e = \begin{bmatrix} 8 & 8 & 8 & 8 \\ 11 & 5 & -5 & -11 \\ 8 & -8 & -8 & 8 \\ 5 & -11 & 11 & -5 \end{bmatrix}$$

$$T_4^o = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

sind.

32. Computer-lesbares Medium, das Computer-ausführbare Instruktionen aufweist, die eingerichtet sind einen Computer zu befähigen, das Verfahren von einem der Ansprüche 1 bis 31 auszuführen.

33. Konverter zum Konvertieren eines Mediendatenblocks, der die Dimensionen n und m aufweist, umfassend:
einen Ersteller zum Erstellen einer Transformationsmatrix, die aus Integerkoeffizienten besteht, die eine zweite Transformationsmatrix, die Nicht-Integerkoeffizienten hat, gemäß vorbestimmter Toleranzen und vorbestimmter Beschränkungen annähert; und
eine Transformationsstufe zum Ausführen einer Transformation durch das Ausführen mindestens einer Matrixmultiplikation des Mediendatenblocks mit der Transformationsmatrix, die aus Integerkoeffizienten besteht, zum Erlangen eines transformierten Mediendatenblocks;
dadurch gekennzeichnet, dass
der Ersteller eingerichtet ist, die Transformationsmatrix durch folgende Schritte zu erstellen:
(a) Ermitteln eines konstanten Faktors aus n- und m-Punkt-DC-Basisfunktionen, die innerhalb einer Toleranz Beschränkungen unterliegen, dass Beträge der n- und m-Punkt-DC-Basisfunktionen übereinstimmen, und dass Basisfunktionen Transformationsbereichsdaten innerhalb eines begrenzten Bereichs von Integerwerten erzeugen;
(b) Ermitteln einer Reihe von konstanten Faktoren aus ungeraden Basisfunktionen von einem grösseren der n- oder m-Punkte, die innerhalb einer Toleranz Beschränkungen unterliegen, dass solche ungeraden Basisfunktionen orthogonal sind, zu entsprechenden konstanten Basisfunktionsfaktoren der zweiten Transformationsmatrix gut korrelieren und mit den DC-Basisfunktionen betragsmässig übereinstimmen; und
(c) Ermitteln einer Reihe von konstanten Faktoren aus geraden Basisfunktionen von dem grösseren der n- oder m-Punkte, die innerhalb einer Toleranz Beschränkungen unterliegen, dass solche geraden Basisfunktionen orthogonal sind, zu entsprechenden konstanten Basisfunktionsfaktoren der zweiten Transformationsmatrix gut korrelieren und mit den DC- und ungeraden Basisfunktionen betragsmässig übereinstimmen.

34. Konverter gemäß Anspruch 33, des Weiteren umfassend:
eine Quantisierungsstufe, die die Wirkung ausübt, den Transformations-Bereichsblock zu quantisieren;
eine Dequantisierungsstufe, die die Wirkung ausübt, den Transformations-Bereichsblock zu dequantisieren; und
eine Invers-Transformationsstufe zum Ausführen einer Invers-Transformation des Transformations-Bereichsblocks, um ein einen rekonstruierten Block zu erzeugen, wobei die Invers-Transformation als eine Sequenz von Matrixmultiplikationen durch eine Transformationsmatrix implementiert ist, die aus Integerzahlen besteht, die innerhalb einer vorgegebenen Toleranz bestimmte Beschränkungen erfüllen, wobei die Beschränkungen eine skalierte Integerbeschränkung, eine perfekte Rekonstruierungsbeschränkung, eine DCT-ähnliche Basisbeschränkung und eine Integer-Bereichslimitierungsbeschränkung umfassen.

Es folgen 6 Blatt Zeichnungen

Anhängende Zeichnungen

Figure 1

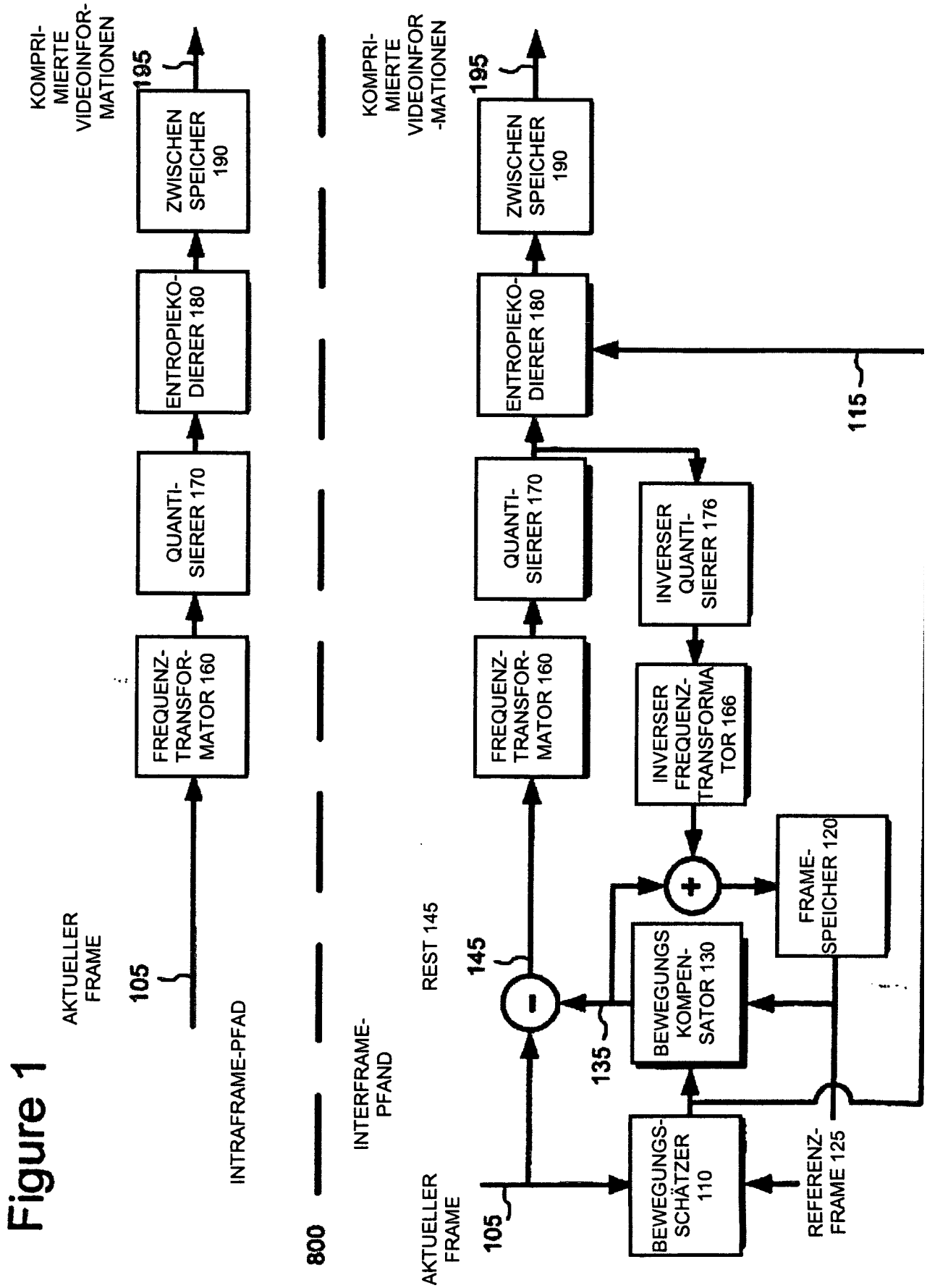

Figure 2

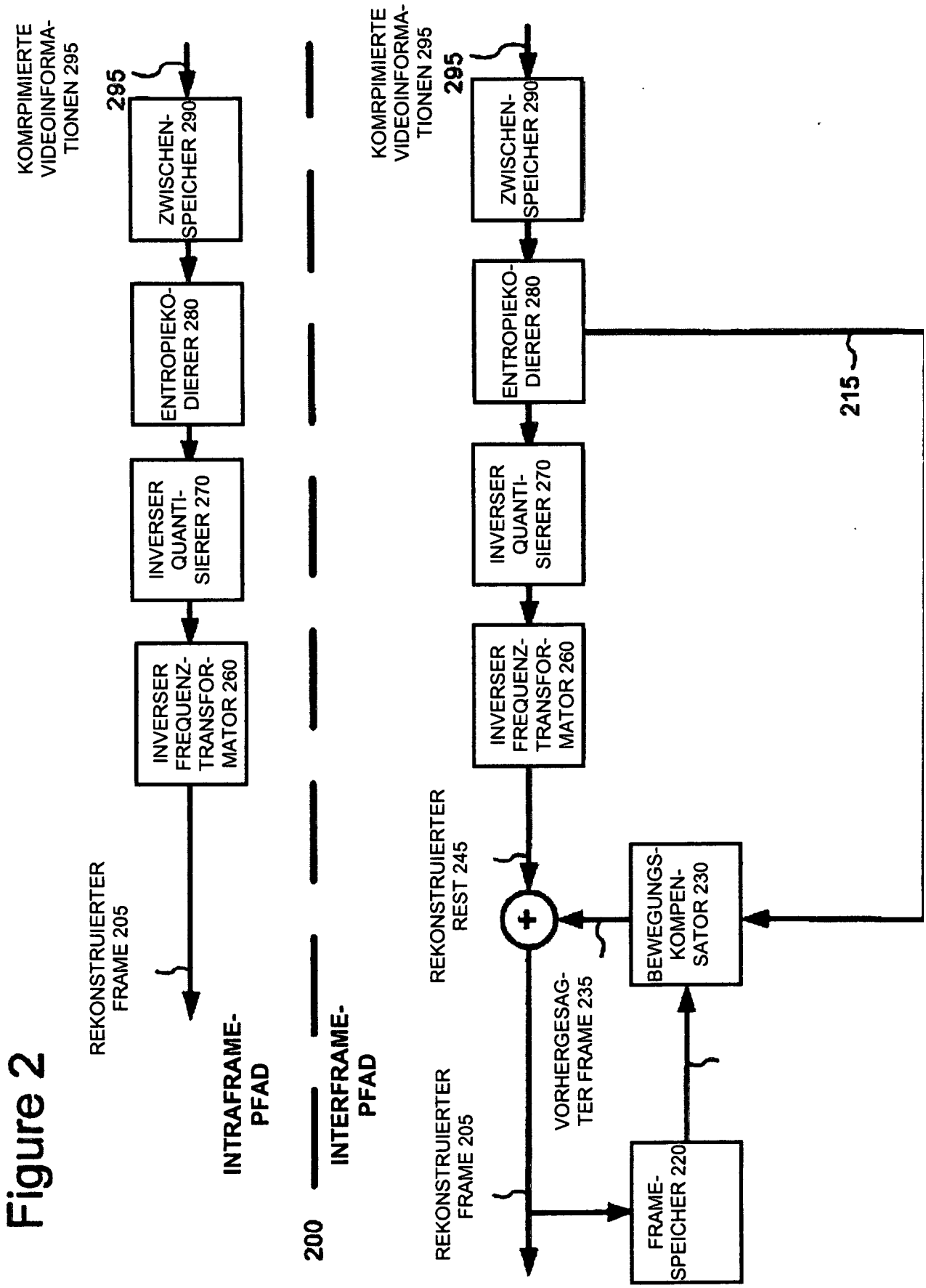
KOMRPIMIERTE VIDEOINFORMA-TIONEN 295
295
ZWISCHEN-SPEICHER 290
ENTROPIEKO-DIERER 280
INVERSER QUANTI-SIERER 270
INVERSER FREQUENZ-TRANSFOR-MATOR 260
REKONSTRUIERTER FRAME 205
INTRAFRAME-PFAD
200
INTERFRAME-PFAD
KOMRPIMIERTE VIDEOINFORMA-TIONEN 295
295
ZWISCHEN-SPEICHER 290
ENTROPIEKO-DIERER 280
215
INVERSER QUANTI-SIERER 270
INVERSER FREQUENZ-TRANSFOR-MATOR 260
REKONSTRUIERTER REST 245
BEWEGUNGS-KOMPEN-SATOR 230
VORHERGESAG-TER FRAME 235
REKONSTRUIERTER FRAME 205
FRAME-SPEICHER 220

Figure 3

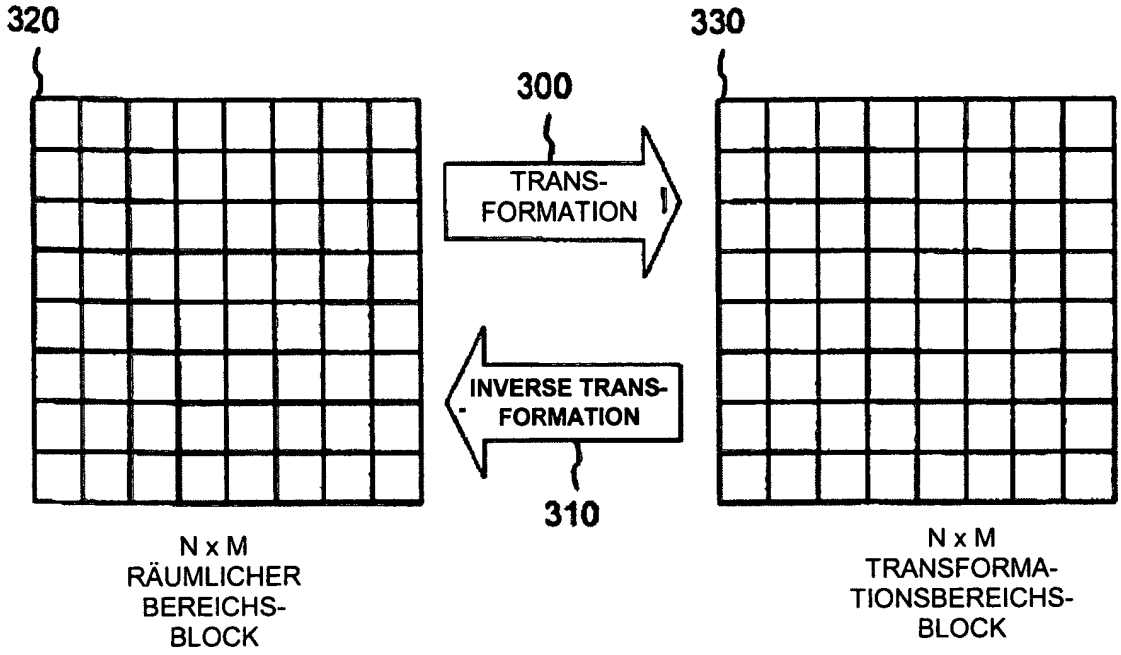

Figure 8

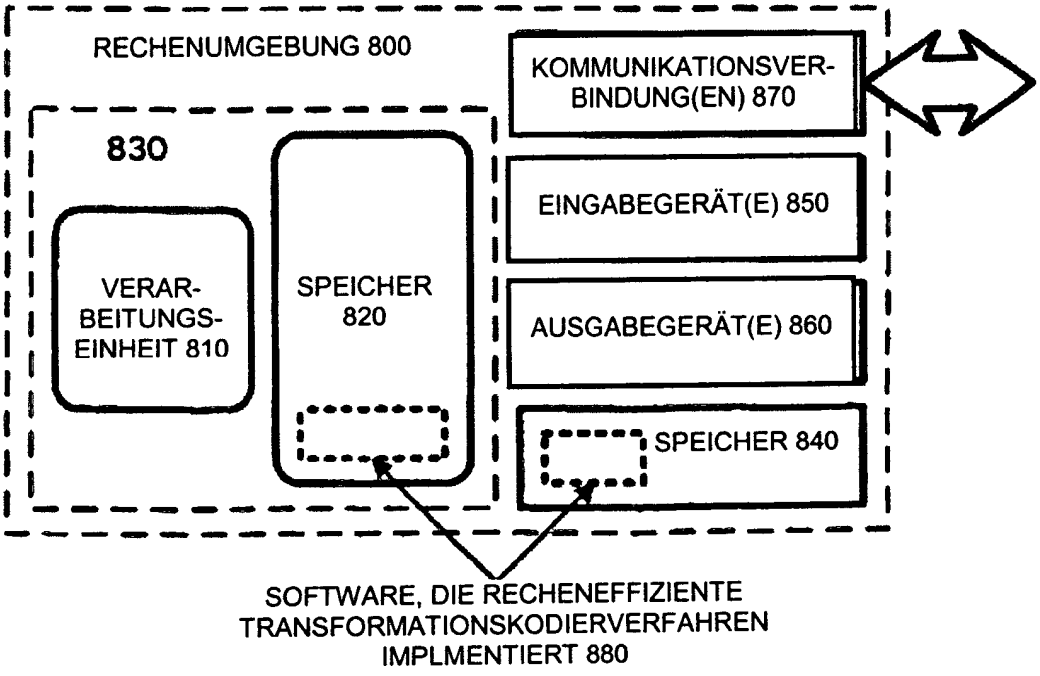

Figure 4

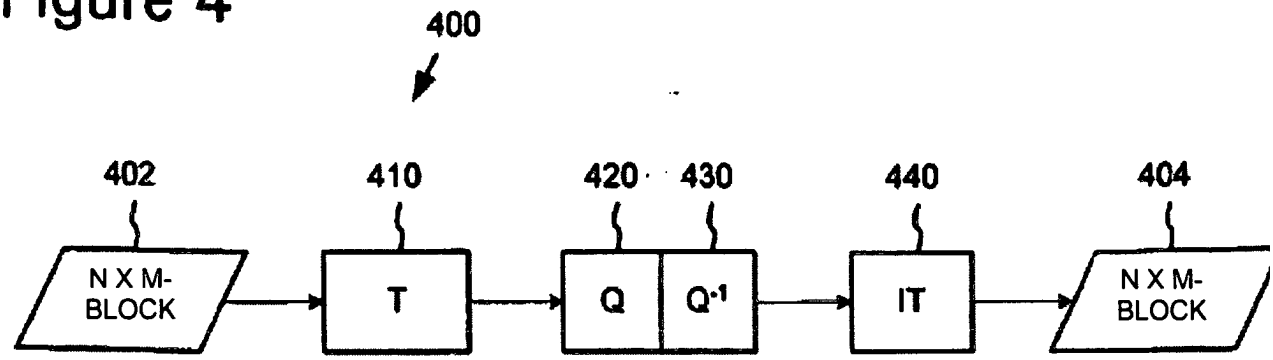

Figure 5

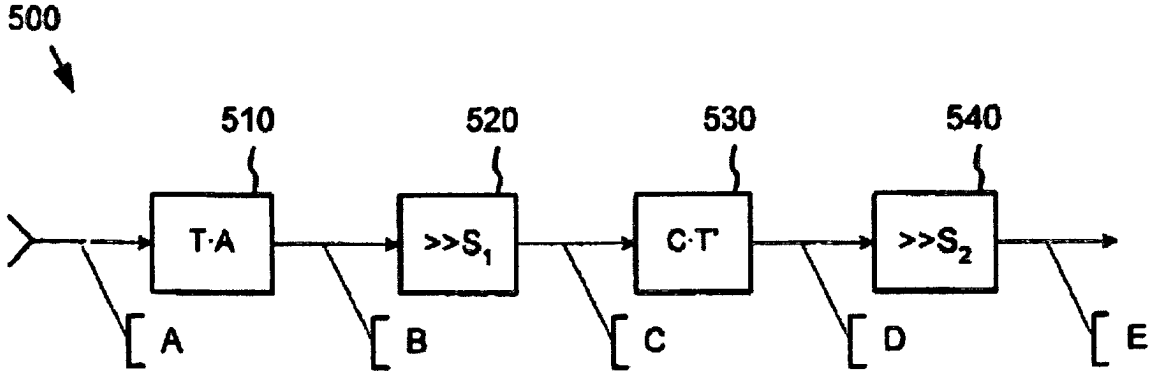

Figure 6

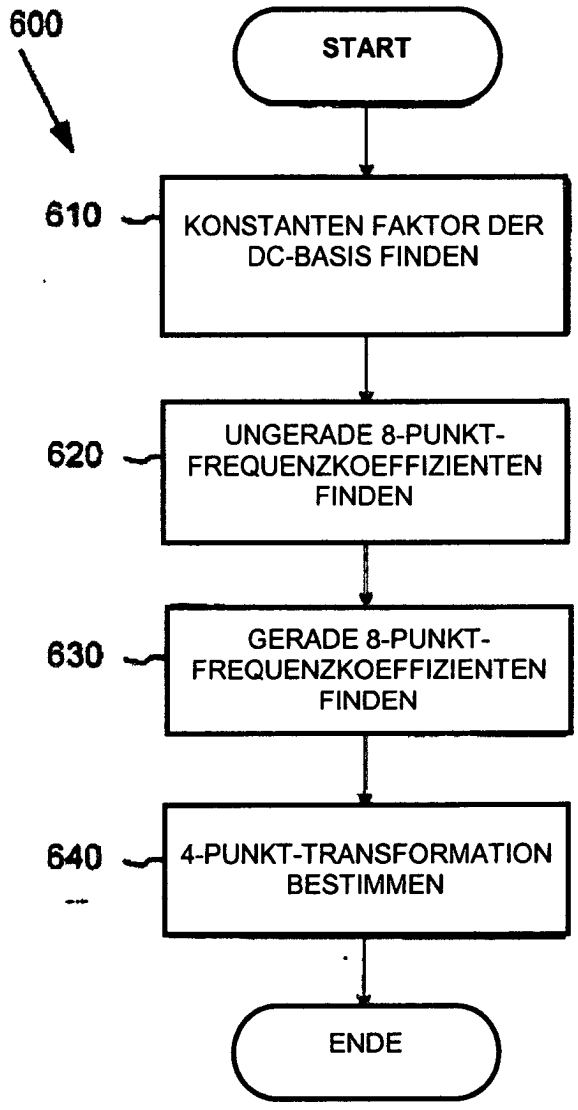
600
START
610
KONSTANTEN FAKTOR DER DC-BASIS FINDEN
620
UNGERADE 8-PUNKT-FREQUENZKOEFFIZIENTEN FINDEN
630
GERADE 8-PUNKT-FREQUENZKOEFFIZIENTEN FINDEN
640
4-PUNKT-TRANSFORMATION BESTIMMEN
ENDE

Figure 7