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

(10) **DE 603 07 354 T2** 2006.12.14

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 359 768 B1**
(21) Deutsches Aktenzeichen: **603 07 354.9**
(96) Europäisches Aktenzeichen: **03 007 020.5**
(96) Europäischer Anmeldetag: **27.03.2003**
(97) Erstveröffentlichung durch das EPA: **05.11.2003**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **09.08.2006**
(47) Veröffentlichungstag im Patentblatt: **14.12.2006**

(51) Int Cl.[8]: ***H04N 7/36*** (2006.01)

(30) Unionspriorität:

371860 P	**10.04.2002**	**US**
379615 P	**04.03.2003**	**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, RO, SE, SI, SK, TR

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

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

(72) Erfinder:
Srinivasan, Sridhar, Seattle, Washington 98109, US; Mukerjee, Kunal, Redmond, Washington 98053, US

(54) Bezeichnung: **Chrominanz-Bewegungsvektorrundung**

Beschreibung

Technisches Gebiet

[0001] Es werden Techniken und Werkzeuge zur Rundung von Chrominanzbewegungsvektoren beschrieben. So findet bei einem Videokodierer/Dekodierer beispielsweise ein Modus (Betriebsart) von mehreren Modi zur Rundung von Chrominanzbewegungsvektoren Anwendung, darunter ein Modus, bei dem die Chrominanzbewegungsvektoren Viertelpixelgenauigkeit aufweisen.

Hintergrund

[0002] Digitale Videos verbrauchen große Mengen an Speicher- und Übertragungskapazitäten. Eine typische unbearbeitete digitale Videosequenz enthält 15 bis 30 Rahmen (frames) pro Sekunde. Jeder Rahmen kann Zehntausende oder Hunderttausende von Pixeln (auch „Pels" genannt) enthalten. Jedes Pixel stellt ein winziges Element eines Bildes dar. In unbearbeiteter Form stellt ein Computer ein Pixel gängigerweise durch 24 Bit dar. So kann ein Pixel beispielsweise einen 8 Bit langen Luminanzwert (auch Luma-Wert genannt), der die Grauskalenkomponente des Pixels festlegt, und zwei 8 Bit lange Chrominanzwerte (auch Chroma-Werte genannt), die die Farbkomponente des Pixels festlegen, enthalten. Damit kann die Anzahl der Bits pro Sekunde, das heißt die Bitrate, einer typischen unbearbeiteten digitalen Videosequenz bei bis zu 5 Millionen Bit pro Sekunde oder mehr liegen.

[0003] Viele Computer und Computernetzwerke verfügen nicht über die Ressourcen, um unbearbeitete digitale Videos zu verarbeiten. Aus diesem Grunde bedienen sich Entwickler der Kompression (auch Kodierung (coding oder encoding) genannt), um die Bitrate des digitalen Videos zu verringern. Die Kompression kann verlustfrei sein, wodurch die Qualität des Videos nicht leidet, der Senkung der Bitrate jedoch durch die Komplexität des Videos Schranken gesetzt sind. Die Kompression kann jedoch auch verlustbehaftet sein, wodurch die Qualität des Videos leidet, die Senkung der Bitrate jedoch drastischer ausfällt. Die Dekompression ist der zur Kompression inverse beziehungsweise umgekehrte Vorgang.

[0004] Zu den Videokompressionstechniken zählen allgemein die Intrarahmenkompression (intraframe compression) und die Interrahmenkompression (interframe compression). Bei den Intrarahmenkompressionstechniken werden einzelne Rahmen, die üblicherweise I-Rahmen oder Schlüsselrahmen (key frames) genannt werden, komprimiert. Bei Interrahmenkompressionstechniken hingegen erfolgt die Kompression von Rahmen in Abhängigkeit von vorhergehenden und/oder nachfolgenden Rahmen, die üblicherweise Prädiktionsrahmen (predicted frame), P-Rahmen oder B-Rahmen genannt werden.

[0005] Das Programm Windows Media Video, Version 8 („WMV8") der Microsoft Corporation enthält einen Videokodierer und einen Videodekodierer. Der WMV8-Kodierer bedient sich der Intrarahmen- und der Interrahmenkornpression, und der WMV8-Dekodierer bedient sich der Intrarahmen- und der Interrahmendekompression. Die Interrahmenkompression in dem WMV8-Kodierer bedient sich einer blockbasierten bewegungsausgeglichenen Prädiktion zur Kodierung, gefolgt von einer Wandlungskodierung des Restfehlers.

[0006] Beim WMV8-Programm wird ein Rahmen in Form von drei Pixelebenen dargestellt: eine Luminanzebene (Y) der Luminanzpixelwerte und zwei Chrominanzebenen (U, V) der Chrominanzpixelwerte. Die Auflösung der Y-Ebene entspricht in der Horizontalen und der Vertikalen dem Doppelten der Auflösung der U- und V-Ebenen. Daher umfasst ein Rahmen mit 320 Pixeln zu 240 Pixeln eine Y-Ebene mit 320 Pixeln zu 240 Pixeln sowie U- und V-Ebenen mit 160 Pixeln zu 120 Pixeln.

[0007] Der WMV8-Kodierer unterteilt einen Prädiktionsrahmen in 8 × 8-Pixelblöcke. Gruppen von vier 8 × 8-Luminanzblöcken und zwei kolokalisierten 8 × 8-Chrominanzblöcken (der eine für die U-Chrominanzebene und der andere für die V-Chrominanzebene) bilden 16 × 16-Makroblöcke. Daher umfasst jeder 16 × 16-Makroblock vier 8 × 8-Luminanzblöcke und zwei 8 × 8-Chrominanzblöcke.

[0008] Für einen Makroblock eines Prädiktionsrahmens nimmt der WMV8-Kodierer eine Bewegungsabschätzung vor. Bei der Bewegungsabschätzung wird die Bewegung eines Makroblocks in einem Prädiktionsrahmen approximiert, indem nach dem Makroblock in dem Prädiktionsrahmen gesucht wird, und der Makroblock in dem Prädiktionsrahmen an einen Makroblock aus einem Referenzrahmen angepasst wird. Gemäß Fig. 1 berechnet beispielsweise der WMV8-Kodierer einen Bewegungsvektor für einen Makroblock (**115**) in dem Prädiktionsrahmen (**110**). Zur Berechnung des Bewegungsvektors sucht der Kodierer in einem Suchbereich (**135**) eines Referenzrahmens (**130**). Innerhalb des Suchbereiches (**135**) vergleicht der Kodierer die Luminanzwerte

des Makroblocks (**115**) aus dem Prädiktionsrahmen (**110**) mit den Luminanzwerten verschiedener Kandidatenblöcke aus dem Referenzrahmen (**130**), um einen passenden Treffer zu finden. Bei dem WMV8-Kodierer kann die Bewegungsvektorgenauigkeit eingestellt werden. Es können zudem ein Suchbereich und Bewegungsvektoren mit einer horizontalen Vollpixel-, Halbpixel- oder Viertelpixelauflösung sowie einer vertikalen Vollpixel- oder Halbpixelauflösung verwendet werden. Mit subpixelgenauen Bewegungsvektoren kann der WMV8-Kodierer die Subpixelbewegung in einer Videosequenz annähern.

[0009] Während des Bewegungsausgleiches bedient sich der WMV8-Kodierer der Bewegungsvektoren für Makroblöcke des Prädiktionsrahmens, um die Prädiktoren für die Makroblöcke aus dem Referenzrahmen zu bestimmen. Für jeden der Bewegungsprädiktionsmakroblöcke berechnet der WMV8-Kodierer die Differenz (auch Rest oder Fehler genannt) zwischen dem ursprünglichen Makroblock und dessen Prädiktor. Der WMV8-Kodierer unterteilt den Rest in Blöcke und komprimiert die restlichen Blöcke verlustbehaftet. Zur Rekonstruktion der Bewegungsprädiktionsmakroblöcke des Prädiktionsrahmens dekomprimiert der WMV8-Kodierer die Reste und addiert sie zu den Prädiktoren für die jeweiligen Makroblöcke.

[0010] Der WMV8-Kodierer bedient sich zudem der Bewegungsvektoren für Makroblöcke der Prädiktionsrahmen, um die Prädiktoren für die Makroblöcke aus dem Referenzrahmen zu bestimmen. Zur Rekonstruktion der Bewegungsprädiktionsmakroblöcke des Prädiktionsrahmens dekomprimiert der WMV8-Dekodierer die Reste und addiert sie zu den Prädiktoren für die Makroblöcke.

[0011] Während der Bewegungsabschätzung oder des Bewegungsausgleiches müssen der WMV8-Kodierer oder der entsprechende Dekodierer, wenn ein Bewegungsvektor eine Subpixelgenauigkeit (das heißt Halbpixel- oder Viertelpixelgenauigkeit) aufweist, Pixelwerte an Subpixelpositionen in dem Referenzrahmen bestimmen. Der WMV8-Kodierer oder der entsprechende Dekodierer erzeugen Werte für Subpixelpositionen unter Verwendung von Interpolationsfiltern. **Fig. 2** zeigt Subpixelabtastpositionen H_0, H_1, H_2, die Werte aufweisen, die mittels Interpolation der Vollpixelwerte a, b, c, ..., p berechnet sind.

[0012] Wird mit Halbpixelbewegungsvektorgenauigkeit gearbeitet, so lauten die Interpolationsfilter, die für Luminanzpixelwerte an drei verschiedenen Halbpixelpositionen H_0, H_1, H_2 verwendet werden, folgendermaßen.

$$H_0 = (f + g + R_2) >> 1 \quad (1)$$

$$H_1 = (f + j + R_2) >> 1 \quad (2)$$

$$H_2 = (f + g + j + k + R_1) >> 2 \quad (3)$$

[0013] Hierbei bezeichnen R_1 und R_2 Rundungssteuerwerte, die von einem 1-Bit-Rundungssteuermerker (flag) gesteuert werden, der die Rundungsart beziehungsweise den Rundungsmodus für einen bestimmten Rahmen angibt. Wird der Rundungssteuermerker auf 0 gesetzt, so gilt $R_1 = 2$ und $R_2 = 1$. Wird der Rundungssteuermerker auf 1 gesetzt, so gilt $R_1 = R_2 = 0$. Der Wert des Rundungssteuermerkers wechselt für jeden P-Rahmen zwischen 1 und 0. Bei jedem I-Rahmen wird der Wert des Rundungssteuermerkers auf 0 zurückgesetzt. Auf diese Weise arbeitet die Rundungssteuerung rahmenweise.

[0014] Gleichungen 1, 2 und 3 stellen Beispiele für eine bilineare Interpolation dar. Die bilineare Interpolation ist schnell und erzeugt tendenziell glatte Pixelwerte. Die Glättung kann gewünschte Effekte (so beispielsweise eine gesenkte Wahrnehmbarkeit des Quantisierungsrauschens) aufweisen, sie kann jedoch auch zu einem Verlust gültiger Pixelinformation führen.

[0015] Bei einer Viertelpixelbewegungsvektorauflösung bedienen sich der WMV8-Kodierer oder der entsprechende Kodierer zunächst bikubischer Filter, um Luminanzpixelwerte an Halbpixelpositionen zu interpolieren. Die bikubische Interpolation ist langsamer als die bilineare Interpolation, behält jedoch tendenziell Randwerte bei und führt zu einem geringeren Verlust gültiger Pixelinformation. Die bikubischen Filter für drei verschiedene Halbpixelpositionen H_0, H_1 und H_2 lauten folgendermaßen.

$$H_0 = (-e + 9f + 9g - h + 8) >> 4 \quad (4)$$

$$H_1 = (-b + 9f + 9j - n + 8) >> 4 \quad (5)$$

$$H_2 = (-t_0 + 9t_1 + 9t_2 - t_3 + 8) » 4 \quad (6)$$

[0016] Hierbei werden t_0, t_1, t_2 und t_3 folgendermaßen berechnet.

$$t_0 = (–a + 9b + 9c – d + 8) >> 4 \quad (7)$$

$$t_1 = (–e + 9f + 9g – h + g) >> 4 \quad (8)$$

$$t_2 = (–i + 9j + 9k – i + 8) >> 4 \quad (9)$$

$$t_3 = (–m + 9n + 9o – p + 8) >> 4 \quad (10)$$

[0017] Die Gleichungen (4) bis (10) können zu einer Ausgabe außerhalb des Bereiches der Eingabewerte führen. So erzeugt beispielsweise bei einer 8 Bit langen Eingabe (Bereich 0 bis 255) die Reihe von Werten 0 255 255 0 in jeder der Gleichungen (4) bis (10) einen Ausgabewert von 287. Daher schneiden (clamp oder clip) der WMV8-Kodierer oder der entsprechende Dekodierer den Ausgabewert jeder der Gleichungen (4) bis (10) derart ab, dass dieser innerhalb des gültigen Bereiches liegt. So werden beispielsweise für 8 Bit lange Ausgabewerte Werte kleiner als 0 zu 0 geändert, während Werte größer als 255 zu 255 geändert werden. Das Abschneiden löst das Bereichsproblem, verlangsamt jedoch die Berechnung. Darüber hinaus führt das Abschneiden zu einer Abnahme der Genauigkeit.

[0018] Der WMV8-Kodierer oder der entsprechende Dekodierer berechnen anschließend Pixelwerte an bestimmten Viertelpixelpositionen in einer nachfolgenden Interpolationsstufe. Diese Viertelpixelstellen sind horizontal zwischen entweder zwei Halbpixelstellen oder einer Vollpixelstelle und einer Halbpixelstelle angeordnet. Mit Blick auf die Viertelpixelstellen bedienen sich der WMV8-Kodierer oder der entsprechende Dekodierer unter Verwendung zweier horizontal benachbarter Halbpixel-/Vollpixelstellen ohne Rundungssteuerung einer bilinearen Interpolation (das heißt (x + y + 1) >> 1).

[0019] Sobald die Luminanzbewegungsvektoren berechnet sind, leiten der WMV8-Kodierer oder der entsprechende Dekodierer die kolokalisierten Chrominanzbewegungsvektoren her. Da die Chrominanzebene beim WMV8-Programm sowohl in der Horizontalen wie auch in der Vertikalen halb so groß wie die Luminanzebene ist, müssen die Luminanzbewegungsvektorwerte zu geeigneten Chrominanzbewegungsvektorwerten skaliert werden. Beim WMV8-Programm beinhaltet dieser Umwandlungsvorgang das Halbieren der Luminanzbewegungsvektoren und das Runden der resultierenden Chrominanzbewegungsvektoren auf Halbpixelgenauigkeit. Auf diese Weise werden Luminanzbewegungsvektoren mit Halbpixelgenauigkeit nicht in Chrominanzbewegungsvektoren mit Viertelpixelgenauigkeit umgewandelt. Darüber hinaus stellt der Vorgang des Chrominanzrundens beim WMV8-Programm einen Single-Mode-Vorgang dar, der von einem Anwender nicht geändert oder ausgewählt werden kann.

[0020] Beim WMV8-Programm können die Pixelwerte an Subpixelpositionen in einem Referenzrahmen unter bestimmten Umständen Bereichsunterschreitungen (underflow) oder Bereichsüberschreitungen (overflow) erfahren. So kann beispielsweise der Luminanzpixelwert an einer Viertelpixelposition 271 (was außerhalb des Bereiches zwischen 0 und 255 ist) sein, wenn der benachbarte Vollpixelpositionswert 255 ist, und der benachbarte Halbpixelpositionswert bei 287 liegt (0 + 9·255 + 9·255 – 0 + 8 >> 4 = 287 und 255 + 287 + 1 >> 1 = 271). Um dieses Problem zu lösen, schneiden nach dem Addieren der Restblöcke zu dem Prädiktor für einen Makroblock der WMV8-Kodierer und der entsprechende Dekodierer gegebenenfalls die rekonstruierten Werte für den Makroblock derart ab, dass sie innerhalb eines Bereiches zwischen 0 und 255 liegen.

[0021] Neben dem WMV8-Programm befassen sich einige weitere internationale Standards mit Videokompression und Videodekompression. Zu diesen Standards zählen MPEG-1, MPEG-2 und MPEG-4 (Motion Picture Experts Group MPEG, Expertengruppe für bewegte Bilder) sowie H.261, H.262 und H.263 der ITU (International Telecommunication Union ITU, Internationale Telekommunikationsunion). Analog zum WMV8-Programm bedienen sich diese Standards einer Kombination aus Intrarahmen- und Interrahmenkompression, wobei sich diese Standards üblicherweise in Details von den beim WMV8-Programm verwendeten Kompressionstechniken unterscheiden.

[0022] Einige Standards (so beispielsweise MPEG-4 und H.263) bedienen sich einer Halbpixelbewegungsabschätzung und eines Halbpixelbewegungsausgleiches unter Verwendung bilinearer Filter und einer Basisrundungssteuerung. Darüber hinaus werden beim H.263-Standard Chrominanzbewegungsvektoren, die theoretisch eine Viertelpixelauflösung (das heißt die Hälfte der Auflösung der Halbpixelluminanzbewegungsvektoren) aufweisen, entweder auf Halbpixelgenauigkeit oder auf Vollpixelgenauigkeit gerundet, sodass keine Viertelpixelwerte im Chrominanzraum zugelassen sind. Für zusätzliche Details betreffend die Bewegungsabschät-

zung und den Bewegungsausgleich bei den genannten Standards ziehe man die Spezifikationen der Standards zu Rate.

[0023] Die europäische Patentanmeldung EP 0 884 912 A2 beschreibt Videokodierer/Dekodierer, bei denen sowohl Frames der Sorte P+ (mit einer Rundung halbzahliger Werte weg von Null) wie auch Frames der Sorte P– (mit einer Rundung halbzahliger Werte hin zu Null) zum Zwecke der Beseitigung von Rundungsfehlern sowie zum Zwecke der Verhinderung eines Anhäufens von Rundungsfehlern zum Einsatz kommen. In dem Beitrag „Very High Efficiency VLSI Chip-Pair for Full Search Block Matching with Fractional Precision" von Kun-Min Yang, veröffentlicht bei Proc. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, Glasgow, UK, 23. bis 26. Mai, 1989, XP 010083029, werden Ausgestaltungen einer VLSI-Architektur sowie Implementierungen eines Chippaares für einen Bewegungsausgleichs-Vollsuche-Blockmatching-Algorithmus beschrieben. Das Patent US 5,659,365 offenbart Chrominanzbewegungsvektoren, die unter Verwendung einer in einem Extrabit enthaltenen Rundungsinformation für Luminanzbewegungsvektoren mit ungeradzahligen Werten berechnet werden. In dem Beitrag „A Motion Compensation Technique for Down-Scaled Pictures in Layered Coding" von Masahiro Iwahashi et al., veröffentlicht bei IEICE Transactions on Communications E77-B (1994), August, Nr. 8, Tokyo, XP 00470652, wird eine Technik zum Bewegungsausgleich beschrieben, bei der ein Drift genanntes Verwischen bei sich bewegenden Bildern gemindert wird, die mittels schichtenbasierter Kodiersysteme niederskaliert sind.

[0024] Die Bewegungsabschätzung und der Bewegungsausgleich stellen effektive Kompressionstechniken dar. Die bislang gängigen Techniken der Bewegungsabschätzung und des Bewegungsausgleiches (so beispielsweise beim WMV8-Programm und den vorgenannten Standards) weisen jedoch einige Nachteile auf, darunter die folgenden Nachteile.

(1) Bei der Berechnung von Pixelwerten an Subpixelpositionen in Referenzrahmen sinkt bei den Kodierern und Dekodierern unnötigerweise die Genauigkeit der vorläufigen Werte. Wird beispielsweise der Pixelwert für eine Viertelpixelposition beim WMV8-Programm berechnet, so werden vorläufige Werte an Halbpixelpositionen um 4 Bit nach rechts verschoben, und dies ungeachtet der Tatsache, dass eine größere Bittiefe zur Verfügung stünde. Darüber hinaus schneiden der WMV8-Kodierer und der entsprechende Dekodierer die vorläufigen Werte während der zweistufigen Interpolation von Viertelpixelpositionen ab, was die Berechnungsgeschwindigkeit senkt, und was zu einer unnötigen Abnahme der Genauigkeit führt.

(2) Die Interpolation für Pixelwerte ist bei der Viertelpixelbewegungsabschätzung und dem Viertelpixelbewegungsausgleich in vielen Fällen ineffizient. So bedingt beispielsweise beim WMV8-Programm die Berechnung einer eindimensionalen Viertelpixelposition die Verwendung eines Filters für eine Halbpixelposition, gefolgt von der Verwendung eines bilinearen Filters.

(3) Die Kodierer und Dekodierer sind nicht in der Lage, auf eine Häufung von Rundungsfehlern geeignet zu reagieren, die bei einer mehrstufigen Interpolation auftreten kann. Rundungsfehler treten beispielsweise dann auf, wenn Pixelwerte in einer Videosequenz von Rahmen zu Rahmen wiederholt abgerundet werden. Dieser Rundungsfehler kann bei Videosequenzen niedriger Qualität und niedriger Bitrate wahrnehmbare Artefakte bewirken. Interpolieren beispielsweise der WMV8-Kodierer und der entsprechende Dekodierer für einen Pixelwert an einer Viertelpixelposition in mehreren Stufen, so kommt keine Rundungssteuerung zum Einsatz. Anstatt dessen werden die Ergebnisse jeder Stufe auf dieselbe Weise in jeder Interpolationsstufe (und ohne Rundungssteuerung) gerundet.

(4) Die Chrominanzrundung erfolgt nicht auf Viertelpixelgenauigkeit, und es besteht keine Kontrolle über die Optionen der Chrominanzbewegungsvektorrundung. So runden beispielsweise der WMV8-Kodierer und der entsprechende Dekodierer alle Chrominanzbewegungsvektoren auf einen Halbpixelwert und können nur in diesem einen Betriebszustand betrieben werden (single mode).

[0025] Eingedenk der kritischen Wichtigkeit der Bewegungsabschätzung und des Bewegungsausgleiches bei digitalen Videos ist nicht überraschend, dass die Bewegungsabschätzung und der Bewegungsausgleich weit fortentwickelte Gebiete darstellen. Wie groß auch immer die Vorteile bereits bestehender Techniken der Bewegungsabschätzung und des Bewegungsausgleiches sind, sie reichen nicht an die Vorteile der nachfolgend beschriebenen Techniken und Werkzeuge heran.

Zusammenfassung

[0026] Die Aufgabe der vorliegenden Erfindung besteht darin, das Leistungsvermögen bei der Umwandlung von Chrominanzbewegungsvektoren bei verschiedenen Anwendungen zu verbessern. Die Aufgabe wird durch die vorliegende Erfindung gemäß Definition in den unabhängigen Ansprüchen gelöst. Ausführungsbeispiele der Erfindung sind in den abhängigen Ansprüchen niedergelegt.

[0027] Die Detailbeschreibung betrifft, zusammengefasst dargestellt, verschiedene Techniken und Werkzeuge zur Rundung von Chrominanzbewegungsvektoren. Zum Bewegungsausgleich bei Videokodier- und Dekodieranwendungen wandelt ein Videokodierer/Dekodierer beispielsweise Bewegungsvektoren im Luminanzraum (Luminanzbewegungsvektoren) in Bewegungsvektoren im Chrominanzraum (Chrominanzbewegungsvektoren) um.

[0028] Gemäß einem Aspekt wandelt eine Komponente, so beispielsweise ein Videokodierer oder ein entsprechender Dekodierer, Luminanzbewegungsvektoren mit Viertelpixelgenauigkeit in Chrominanzbewegungsvektoren mit Viertelpixelgenauigkeit um. So werden beispielsweise Luminanzbewegungsvektoren mit Halbpixelwerten im Luminanzraum in Chrominanzbewegungsvektoren mit entsprechenden Viertelpixelwerten im Chrominanzraum umgewandelt. Die Verwendung viertelpixelgenauer Chrominanzbewegungsvektoren kann die Qualität der Prädiktion (Vorherbestimmung) beim Bewegungsaugleich verbessern.

[0029] Gemäß einem weiteren Aspekt wandelt eine Komponente, so beispielsweise ein Videokodierer oder ein entsprechender Dekodierer, Luminanzbewegungsvektoren in einem Modus von mehreren verfügbaren Modi zur Rundung von Chrominanzbewegungsvektoren in Chrominanzbewegungsvektoren um. So senkt beispielsweise im Vergleich zu einem ersten Modus ein zweiter Modus die Genauigkeit der Chrominanzbewegungsvektoren, wodurch die Komplexität der Berechnung beim nachfolgenden Bewegungsausgleich verringert wird. Die Verwendung mehrerer Modi zur Rundung von Chrominanzbewegungsvektoren kann die Qualität beim Bewegungsausgleich auf Kosten der Komplexität der Berechnung steigern und umgekehrt.

[0030] Die verschiedenartigen Techniken und Werkzeuge können in Kombination oder unabhängig voneinander verwendet werden. Zusätzliche Merkmale und Vorteile erschließen sich aus der nachfolgenden Detailbeschreibung, die zusammen mit der begleitenden Zeichnung zu betrachten ist.

Kurzbeschreibung der Zeichnung

[0031] Fig. 1 ist ein Diagramm, das die Bewegungsabschätzung in einem Videokodierer entsprechend dem Stand der Technik erläutert.

[0032] Fig. 2 ist ein Diagramm, das die Subpixelstellen für die Interpolation bei der Subpixelbewegungsabschätzung und dem Subpixelbewegungsausgleich entsprechend dem Stand der Technik erläutert.

[0033] Fig. 3 ist ein Blockdiagramm einer geeigneten Berechnungsumgebung, in der mehrere beschriebene Ausführungsbeispiele implementiert sein können.

[0034] Fig. 4 ist ein Blockdiagramm eines verallgemeinerten Videokodiersystems, das bei mehreren beschriebenen Ausführungsbeispielen zum Einsatz kommt.

[0035] Fig. 5 ist ein Blockdiagramm eines verallgemeinerten Videodekodiersystems, das bei mehreren beschriebenen Ausführungsbeispielen zum Einsatz kommt.

[0036] Fig. 6 ist ein Diagramm, das Stellen für die Pixelwertinterpolation während der Subpixelbewegungsabschätzung und des Subpixelbewegungsausgleiches zeigt.

[0037] Fig. 7 ist ein Diagramm, das Vollpixelstellen mit Vollpixeln zeigt, die zur Berechnung interpolierter Pixelwerte für Subpixelstellen verwendet werden.

[0038] Fig. 8 ist ein Diagramm, das eine zweistufige Interpolationstechnik zur Interpolation von Werten an Subpixelpositionen zeigt.

[0039] Fig. 9 ist ein Diagramm, das eine Abtastposition mit einem Halbpixel in der Horizontalen und einem Halbpixel in der Vertikalen sowie die vorläufigen Werte an Subpixelpositionen zeigt, die zur Berechnung des Wertes an der Abtastposition verwendet werden.

[0040] Fig. 10 ist ein Diagramm, das eine Abtastposition mit einem Viertelpixel in der Horizontalen und einem Halbpixel in der Vertikalen, eine Abtastposition mit einem Halbpixel in der Horizontalen und einem Viertelpixel in der Vertikalen und die vorläufigen Werte an Subpixelpositionen zeigt, die zur Berechnung des Wertes an der Abtastposition verwendet werden.

[0041] Fig. 11 ist ein Diagramm, das eine Abtastposition mit einem Viertelpixel in der Horizontalen und einem Viertelpixel in der Vertikalen und die vorläufigen Werte an Subpixelpositionen zeigt, die zur Berechnung des Wertes an der Abtastposition verwendet werden.

[0042] Fig. 12 ist ein Diagramm, das eine mehrstufige Interpolationstechnik mit verbessertem Wertebereich (in Bit) für die vorläufigen Werte zeigt.

[0043] Fig. 13 ist ein Diagramm, das eine mehrstufige Interpolationstechnik mit ausgelassenem Abschneiden zeigt.

[0044] Fig. 14 ist ein Diagramm, das eine mehrstufige Interpolationstechnik mit verschobenem Bitschieben zeigt.

[0045] Fig. 15 ist ein Diagramm, das eine mehrstufige Interpolationstechnik unter Verwendung einer stufenweise wechselnden Rundungssteuerung zeigt.

[0046] Fig. 16 ist ein Flussdiagramm, das eine Technik zum Auswählen zwischen mehreren Betriebsarten beziehungsweise Modi der Chrominanzrundung und Interpolation zeigt.

[0047] Fig. 17 ist eine Tabelle, die eine erste Betriebsart der Chrominanzrundung zeigt.

[0048] Fig. 18 ist eine Tabelle, die eine zweite Betriebsart der Chrominanzrundung zeigt.

Detailbeschreibung

[0049] Die beschriebenen Ausführungsbeispiele betreffen Techniken und Werkzeuge zur Subpixelinterpolation bei der Bewegungsabschätzung und beim Bewegungsausgleich. Verschiedene Ausführungsbeispiele betreffen Techniken und Werkzeuge zum Erhalten der Genauigkeit bei einer mehrstufigen Interpolation mittels Verschiebens des Abschneidens und/oder Bitschiebens (Vorgänge, die zu einer Abnahme der Genauigkeit führen können) bis in spätere Stufen der Interpolation. Weitere Ausführungsbeispiele betreffen effiziente Filter sowie Rundungsverfahren für die mehrstufige Interpolation.

[0050] Ein Kodierer beziehungsweise ein Dekodierer nehmen eine Subpixelinterpolation an einem Referenzrahmen oder an einem Abschnitt eines Rahmens, so beispielsweise an einem oder mehreren Blöcken oder Makroblöcken, vor. Der Kodierer/Dekodierer berechnet Pixelwerte an Subpixelstellen innerhalb des Referenzrahmens. Der Kodierer/Dekodierer kann anschließend einen Bewegungsausgleich unter Verwendung subpixelgenauer Bewegungsvektoren vornehmen.

[0051] Bei einigen Ausführungsbeispielen nehmen ein Videokodierer beziehungsweise ein Videodekodierer die Subpixelinterpolation in einer Videokodieranwendung oder einer Videodekodieranwendung vor. Alternativ nimmt ein anderer Kodierer oder Dekodierer oder eine andere Art von Komponente die Subpixelinterpolation oder die nachstehend beschriebene andere Technik bei einer anderen Art von Anwendung vor.

[0052] Als Alternative zur Durchführung einer Subpixelinterpolation an Referenzrahmen nimmt bei einigen Ausführungsbeispielen ein Kodierer/Dekodierer die Subpixelinterpolation an Feldern, Objektschichten (object layers) oder anderen Bildern vor.

[0053] Bei einigen Ausführungsbeispielen wird die Subpixelinterpolation durch Berechnung von Pixelwerten in den Luminanz- und Chrominanzebenen einer Referenzebene im YUV-Farbraum vorgenommen. Alternativ kann auch ein anderer Farbraum (beispielsweise YIQ oder RGB) verwendet werden.

[0054] Die verschiedenartigen Techniken und Werkzeuge können in Kombination oder unabhängig voneinander verwendet werden. Verschiedene Ausführungsbeispiele implementieren eine oder mehrere der beschriebenen Techniken und eines oder mehrere der beschriebenen Werkzeuge. Obwohl die einzelnen Verfahrensschritte bei diesen Techniken üblicherweise aus Gründen einfacherer Darstellung in einer bestimmten sequenziellen Reihenfolge beschrieben werden, sollte einsichtig sein, dass diese Art der Beschreibung bestimmte Umordnungen der Reihenfolge der Verfahrensschritte einschließt, es sei denn, eine bestimmte Reihenfolge ist explizit gefordert. So können Verfahrensschritte, die nacheinander beschrieben werden, in einigen Fällen umgeordnet oder gleichzeitig ausgeführt werden. Darüber hinaus zeigen Flussdiagramme aus Gründen der einfacheren Darstellung üblicherweise nicht verschiedene Realisierungswege, durch die bestimmte Techniken

in Verbindung mit anderen Techniken in der Praxis umgesetzt werden können.

[0055] Bei einigen Ausführungsbeispielen bedienen sich ein Videokodierer und ein entsprechender Dekodierer verschiedener Merker (flags) und Signale in einem Bitstrom. Obwohl bestimmte Merker und Signale beschrieben werden, sollte einsichtig sein, dass diese Art der Beschreibung andere Konventionen (beispielsweise Nullen anstelle von Einsen) für die Merker und Signale einschließt.

1. Berechnungsumgebung

[0056] Fig. 3 zeigt ein verallgemeinertes Beispiel einer geeigneten Berechnungsumgebung (**300**), durch die mehrere der beschriebenen Ausführungsbeispiele implementiert sein können. Die Berechnungsumgebung (**300**) ist nicht dafür gedacht, irgendeine Beschränkung mit Blick auf den Verwendungs- und Funktionalitätsumfang nahezulegen, da die Techniken und Werkzeuge in verschiedenen Allzweck- oder Sonderzweckberechnungsumgebungen implementiert sein können.

[0057] Wie in Fig. 3 gezeigt ist, umfasst die Berechnungsumgebung (**300**) wenigstens eine Verarbeitungseinheit (**310**) und einen Arbeitsspeicher (**320**). In Fig. 3 ist die grundlegende Ausgestaltung (**330**) mit einer gestrichelten Linie umrandet. Die Verarbeitungseinheit (**310**) führt computerausführbare Anweisungen aus und kann ein reeller oder ein virtueller Prozessor sein. In einem Multiverarbeitungssystem (multi-processing system) führen mehrere Verarbeitungseinheiten computerausführbare Anweisungen aus, um die Verarbeitungsleistung zu erhöhen. Der Arbeitsspeicher (**320**) kann ein flüchtiger Arbeitsspeicher (so beispielsweise ein Register, ein Cache, ein RAM), ein nichtflüchtiger Arbeitsspeicher (so beispielsweise ein ROM, ein EEPROM, ein Flash-Arbeitsspeicher und dergleichen) oder eine Kombination von beidem sein. Der Arbeitsspeicher (**320**) speichert Software (**380**), die die Subpixelinterpolationstechniken in einem Kodierer und/oder Dekodierer implementiert, so beispielsweise als Videokodierer und/oder Dekodierer.

[0058] Die Berechnungsumgebung kann zusätzliche Merkmale aufweisen. So umfasst die Berechnungsumgebung (**300**) beispielsweise einen Speicher (**340**), eine oder mehrere Eingabevorrichtungen (**350**), eine oder mehrere Ausgabevorrichtungen (**360**) sowie eine oder mehrere Datenaustauschverbindungen (**370**). Ein Verbindungsmechanismus (nicht gezeigt), so beispielsweise ein Bus, ein Controller oder ein Netzwerk, verbindet die Komponenten der Berechnungsumgebung (**300**). Typischerweise stellt eine Betriebssystemsoftware (nicht gezeigt) eine Arbeitsumgebung für andere in der Berechnungsumgebung (**300**) arbeitende Software bereit und koordiniert darüber hinaus die Aktivitäten der Komponenten der Berechnungsumgebung (**300**).

[0059] Der Speicher (**340**) kann entfernbar oder nichtentfernbar sein und umfasst magnetische Platten, magnetische Bänder oder Kassetten, CD-ROMs, DVDs und anderen Medien, die zur Speicherung von Information verwendet werden können, und auf die innerhalb der Berechnungsumgebung (**300**) zugegriffen werden kann. Der Speicher (**340**) speichert Anweisungen für die Software (**380**), die die Subpixelinterpolationstechniken implementiert.

[0060] Die Eingabevorrichtung (**350**) beziehungsweise die Eingabevorrichtungen können eine berührungsempfindliche Eingabevorrichtung sein, so beispielsweise eine Tastatur, eine Maus, ein Pen oder ein Trackball, eine stimmbasierte Eingabevorrichtung, eine Abtastvorrichtung oder eine andere Vorrichtung, die die Eingabe in die Berechnungsumgebung (**300**) bewirkt. Zur Audio- oder Videokodierung können die Eingabevorrichtung beziehungsweise die Eingabevorrichtungen (**350**) eine Soundkarte, eine Videokarte, eine TV-Tunerkarte oder ähnliche Vorrichtungen sein, die Audio- oder Videoeingaben in analoger oder digitaler Form annehmen, oder eine CD-ROM oder eine CD-RW, die Audio- oder Videoabtastungen in der Berechnungsumgebung (**300**) ausliest. Die Ausgabevorrichtung (**360**) beziehungsweise die Ausgabevorrichtungen können eine Anzeige, ein Drucker, ein Lautsprecher, ein CD-Schreiber oder eine andere Vorrichtung sein, die die Ausgabe aus der Berechnungsumgebung (**300**) vornehmen.

[0061] Die Datenaustauschverbindung (**370**) beziehungsweise die Datenaustauschverbindungen ermöglichen einen Datenaustausch über ein Datenaustauschmedium zu einer weiteren Berechnungseinheit. Das Datenaustauschmedium liefert Information, so beispielsweise computerausführbare Anweisungen, Audio- oder Videoeingaben und -ausgaben oder andere Daten in Form eines modulierten Datensignals. Ein moduliertes Datensignal ist ein Signal, bei dem eine oder mehrere Eigenschaften derart gesetzt oder geändert sind, dass Information in dem Signal kodiert ist. Als Beispiel und keinesfalls als Beschränkung seien als Datenaustauschmedium unter anderem drahtgebundene oder drahtlose Techniken genannt, die auf einem elektrischen, optischen, hochfrequenztechnischen, infrarottechnischen, akustischen oder anderem Träger implementiert sind.

[0062] Die Techniken und Werkzeuge können im allgemeinen Kontext computerlesbarer Medien beschrieben sein. Computerlesbare Medien sind beliebige verfügbare Medien, auf die innerhalb einer Berechnungsumgebung zugegriffen werden kann. Beispielhalber und keineswegs als Beschränkung seien als computerlesbare Medien in der Berechnungsumgebung (**300**) unter anderem ein Arbeitsspeicher (**320**), ein Speicher (**340**), Datenaustauschmedien und Kombinationen hiervon genannt.

[0063] Die Techniken und Werkzeuge können in dem allgemeinen Kontext computerausführbarer Anweisungen beschrieben sein, so beispielsweise derjenigen, die in Programmmodulen enthalten sind, die in einer Berechnungsumgebung in einem reellen oder virtuellen Zielprozessor ausgeführt werden. Allgemein zählen zu den Programmmodulen Routinen, Programme, Bibliotheken, Objekte, Klassen, Komponenten, Datenstrukturen und dergleichen, die bestimmte Aufgaben ausführen oder bestimmte abstrakte Datentypen implementieren. Die Funktionalität der Programmmodule kann je nach Wunsch bei verschiedenartigen Ausführungsbeispielen kombiniert oder unter bestimmten Programmmodulen aufgeteilt werden. Computerausführbare Anweisungen für Programmmodule können in einer lokalen oder verteilten Berechnungsumgebung ausgeführt werden.

[0064] Im Sinne der vorliegenden Darstellung werden in der Detailbeschreibung Begriffe wie „bestimmen" und „auswählen" zur Beschreibung von Verfahrensschritten auf einem Computer in einer Berechnungsumgebung verwendet. Diese Begriffe stellen hochgradige Abstraktionen für Verfahrensschritte dar, die von einem Computer ausgeführt werden, und dürfen nicht mit Handlungen verwechselt werden, die von einem menschlichen Wesen ausgeführt werden. Die tatsächlichen von einem Computer ausgeführten Verfahrensschritte, die diesen Begriffen entsprechen, ändern sich in Abhängigkeit von der jeweiligen Implementierung.

II. Verallgemeinerte Videokodierer und Videodekodierer

[0065] Fig. 4 ist ein Blockdiagramm eines verallgemeinerten Videokodierers (**400**), während Fig. 5 ein Blockdiagramm eines verallgemeinerten Videodekodierers (**500**) ist.

[0066] Die Beziehungen, die zwischen den Modulen innerhalb des Kodierers und des Dekodierers gezeigt sind, bezeichnen den Hauptinformationsfluss in dem Kodierer und dem Dekodierer. Andere Beziehungen sind aus Gründen einer einfacheren Darstellung nicht gezeigt. Insbesondere sind in Fig. 4 und Fig. 5 keine Zusatzinformationen gezeigt, durch die Kodiereinstellungen, Betriebsarten, Tabellen und dergleichen angegeben werden, die für die Videosequenzen, Rahmen, Makroblöcke, Blöcke oder dergleichen verwendet werden. Derartige Zusatzinformationen werden üblicherweise in dem Ausgabebitstrom gesendet, nachdem eine Entropiekodierung der Zusatzinformation erfolgt ist. Das Format des ausgegebenen Bitstromes kann das Windows-Media-Video-Format oder ein beliebiges anderes Format sein.

[0067] Der Kodierer (**400**) und der Dekodierer (**500**) sind blockbasiert und verwenden das 4:2:0-Makroblockformat, wobei jeder Makroblock vier Luminanz-8 × 8-Luminanz-Blöcke (die bisweilen auch als 16 × 16-Makroblock behandelt werden) und zwei 8 × 8-Chrominanz-Blöcke (beispielsweise einen U-Block und einen V-Block) enthalten. Alternativ sind der Kodierer (**400**) und der Dekodierer (**500**) objektbasiert, verwenden ein anderes Makroblock- oder Blockformat oder führen Operationen an Mengen von Pixeln mit einer Größe oder Ausgestaltung aus, die nicht 8 × 8-Blöcken oder 16 × 16-Blöcken entsprechen.

[0068] In Abhängigkeit von der Implementierung und dem Typ der gewünschten Kompression können Module des Kodierers oder Dekodierers hinzugefügt, weggelassen, auf mehrere Module verteilt, mit anderen Modulen kombiniert und/oder gegen andere Module ausgetauscht werden. Bei alternativen Ausführungsbeispielen bedienen sich Kodierer oder Dekodierer mit verschiedenen Modulen und/oder anderen Ausgestaltungen von Modulen einer oder mehrerer der beschriebenen Techniken.

A. Videokodierer

[0069] Fig. 4 ist ein Blockdiagramm eines allgemeinen Videokodiersystems (**14**). Das Kodiersystem (**400**) empfängt eine Sequenz von Videorahmen mit einem aktuellen Rahmen (**405**) und erzeugt komprimierte Videoinformationen (**495**) als Ausgabe. Bestimmte Ausführungsbeispiele des Videokodierers bedienen sich üblicherweise einer Abwandlung oder einer ergänzten Ausgestaltung des verallgemeinerten Kodierers (**400**).

[0070] Das Kodiersystem (**400**) komprimiert Prädiktionsrahmen und Schlüsselrahmen. Aus Gründen einer einfacheren Darstellung zeigt Fig. 4 einen Weg für Schlüsselrahmen durch das Kodiersystem (**400**) und einen Weg für Vorwärtsprädiktionsrahmen. Viele der Komponenten des Kodiersystems (**400**) werden zur Kompres-

sion sowohl der Schlüsselrahmen wie auch der Prädiktionsrahmen verwendet. Die genauen Verfahrensschritte, die von diesen Komponenten ausgeführt werden, können in Abhängigkeit von der Art der zu komprimierenden Information variieren.

[0071] Ein Prädiktionsrahmen (auch p-Rahmen, b-Rahmen für bidirektionale Prädiktion oder interkodierter Rahmen (intercoded frame) genannt) ermittelt die Prädiktion (beziehungsweise Differenz) zu einem oder mehreren weiteren Rahmen. Der Prädiktionsrest ist die Differenz zwischen der Prädiktion und dem ursprünglichen Rahmen. Demgegenüber wird ein Schlüsselrahmen (auch i-Rahmen oder intrakodierter Rahmen (intracoded frame) genannt) ohne Referenz gegenüber anderen Rahmen komprimiert.

[0072] Ist der aktuelle Rahmen (**405**) ein Vorwärtsprädiktionsrahmen, so schätzt ein Bewegungsabschätzer (**410**) die Bewegung der Makroblöcke oder anderer Mengen von Pixeln des aktuellen Rahmens (**405**) in Bezug auf einen Referenzrahmen ab, der der rekonstruierte vorherige Rahmen (**425**) ist, der in dem Rahmenspeicher (**420**) gespeichert ist. Bei alternativen Ausführungsbeispielen ist der Referenzrahmen ein späterer Rahmen, oder es erfolgt eine bidirektionale Prädiktion des aktuellen Rahmens. Der Bewegungsabschätzer (**410**) gibt als Zusatzinformation eine Bewegungsinformation (**415**) aus, so beispielsweise Bewegungsvektoren. Ein Bewegungsausgleicher (**430**) wendet die Bewegungsinformation (**415**) auf den rekonstruierten vorhergehenden Rahmen (**425**) an, um einen bewegungsausgeglichenen aktuellen Rahmen (**435**) zu bilden. Die Prädiktion ist jedoch selten vollkommen, und die Differenz zwischen dem bewegungsausgeglichenen aktuellen Rahmen (**435**) und dem ursprünglichen aktuellen Rahmen (**405**) ist der Prädiktionsrest (**445**). Alternativ setzen der Bewegungsabschätzer und der Bewegungsausgleicher eine andere Art von Bewegungsabschätzung beziehungsweise Bewegungsausgleich ein.

[0073] Ein Frequenzwandler (**460**) wandelt die Raumdomänenvideoinformation in Frequenzdomänendaten (spektral) um. Bei blockbasierten Videorahmen wendet der Frequenzwandler (**460**) eine diskrete Kosinustransformation (discrete cosine transform DCT) oder eine Variante hiervon auf Blöcke der Bewegungsprädiktionsdaten an, wodurch Blöcke aus DCT-Koeffizienten gebildet werden. Alternativ wendet der Frequenzwandler (**460**) eine andere herkömmliche Frequenzwandlung, so beispielsweise eine Fouriertransformation, an, oder er setzt eine Wavelet- oder Subbandanalyse an. Bei einigen Ausführungsbeispielen wendet der Frequenzwandler (**460**) eine Frequenzumwandlung auf Blöcke von Raumprädiktionsresten für Schlüsselrahmen an. Der Frequenzwandler (**460**) kann eine Frequenzwandlung der Größe 8 × 8, 8 × 4, 4 × 8 oder auch einer anderen Größe vornehmen.

[0074] Ein Quantisierer (**470**) quantisiert anschließend die Blöcke der Spektraldatenkoeffizienten. Der Quantisierer wendet eine gleichmäßige skalare Quantisierung auf die Spektraldaten mit einer Schrittweite an, die in Abhängigkeit von den Rahmen oder auf anderer Basis variiert. Alternativ wendet der Quantisierer eine andere Art von Quantisierung auf die Spektraldatenkoeffizienten an, so beispielsweise eine nichtgleichmäßige, vektorbasierte oder nichtadaptive Quantisierung, oder er quantisiert die Raumdomänendaten in einem Kodiersystem, in dem keine Frequenzwandlung vorgenommen wird, direkt. Zusätzlich zu der adaptiven Quantisierung kann der Kodierer (**400**) die Techniken der Rahmenaussonderung (frame dropping), des adaptiven Filterns (adaptive filtering) oder andere Techniken zur Steuerung der Rate einsetzen.

[0075] Wird ein rekonstruierter aktueller Rahmen für eine nachfolgende Bewegungsabschätzung und einen nachfolgenden Bewegungsausgleich verwendet, so nimmt ein Umkehrquantisierer (**476**) eine Umkehrquantisierung an den quantisierten Spektraldatenkoeffizienten vor. Ein Umkehrfrequenzwandler (**466**) nimmt sodann die Umkehrung der Verfahrensschritte des Frequenzwandlers (**460**) vor, wodurch er einen rekonstruierten Prädiktionsrest (für einen Prädiktionsrahmen) oder einen rekonstruierten Schlüsselrahmen erzeugt. War der aktuelle Rahmen (**405**) ein Schlüsselrahmen, so wird der rekonstruierte Schlüsselrahmen aus dem rekonstruierten aktuellen Rahmen (nicht gezeigt) genommen. War der aktuelle Rahmen (**405**) ein Prädiktionsrahmen, so wird der rekonstruierte Prädiktionsrest zu dem bewegungsausgeglichenen aktuellen Rahmen (**435**) hinzuaddiert, um den rekonstruierten aktuellen Rahmen zu bilden. Der Rahmenspeicher (**420**) puffert den rekonstruierten aktuellen Rahmen zur Verwendung bei der Prädiktion des nächsten Rahmens. Bei einigen Ausführungsbeispielen wendet der Kodierer einen Blockierfilter auf den rekonstruierten Rahmen an, um Diskontinuitäten in den Blöcken des Rahmens aktiv zu glätten.

[0076] Der Entropiekodierer (**480**) komprimiert die Ausgabe des Quantisierers (**470**) wie auch bestimmte Zusatzinformationen (beispielsweise die Bewegungsinformation (**415**), die Größe des Quantisierungsschrittes). Typische Entropiekodiertechniken sind unter anderem das arithmetische Kodieren, das differentielle Kodieren, das Huffmann-Kodieren, das Lauflängenkodieren (run length coding), das LZ-Kodieren, das Wörterbuchkodieren sowie Kombinationen hiervon. Der Entropiekodierer (**480**) verwendet üblicherweise verschiedene Kodier-

techniken für verschiedene Arten von Information (beispielsweise DC-Koeffizienten, AC-Koeffizienten, verschiedene Arten von Zusatzinformation) und kann unter mehreren Kodiertabellen innerhalb einer bestimmten Kodiertechnik wählen.

[0077] Der Entropiekodierer (**480**) verbringt die komprimierte Videoinformation (**495**) in den Puffer (**490**). Ein Pufferfüllstandsanzeiger wird den bitratenadaptiven Modulen zugeleitet. Die komprimierte Videoinformation (**495**) wird dem Puffer (**490**) mit einer konstanten oder vergleichsweise konstanten Bitrate entnommen und zur nachfolgenden Streamingverarbeitung mit jener Bitrate gespeichert. Alternativ nimmt das Kodiersystem (**400**) ein Streaming der komprimierten Videoinformationen unmittelbar im Gefolge der Kompression vor.

[0078] Vor oder nach dem Puffer (**490**) kann die komprimierte Bildinformation (**495**) zum Zwecke der Übertragung über ein Netzwerk kanalkodiert werden. Die Kanalkodierung kann eine Fehlererfassung und Berichtigungsdaten auf die komprimierte Videoinformation (**495**) anwenden.

B. Videodekodierer

[0079] Fig. 5 ist ein Blockdiagramm eines allgemeinen Videodekodiersystems (**500**). Das Dekodiersystem (**500**) empfängt Information (**595**) für eine komprimierte Sequenz von Videorahmen und erzeugt eine Ausgabe mit einem rekonstruierten Rahmen (**505**). Bestimmte Ausführungsbeispiele von Videodekodierern verwenden üblicherweise Abwandlungen oder Ergänzungen des allgemeinen Dekodierers (**500**).

[0080] Das Dekodiersystem (**500**) dekomprimiert Prädiktionsrahmen und Schlüsselrahmen. Aus Gründen einer einfacheren Darstellung zeigt Fig. 5 einen Weg für Schlüsselrahmen durch das Dekodiersystem (**500**) und einen Weg für Vorwärtsprädiktionsrahmen. Viele der Komponenten des Dekodiersystems (**500**) werden zur Dekomprimierung sowohl von Schlüsselrahmen wie auch von Prädiktionsrahmen verwendet. Die genauen Verfahrensschritte, die von diesen Komponenten ausgeführt werden, können von der Art der zu komprimierenden Information abhängen.

[0081] Ein Puffer (**590**) empfängt die Informationen (**595**) für die komprimierte Videosequenz und stellt die empfangenen Informationen einem Entropiedekodierer (**580**) zur Verfügung. Der Puffer (**590**) empfängt die Information üblicherweise mit einer Rate, die über die Zeit vergleichsweise konstant ist, und enthält einen Flimmerpuffer (jitter buffer), um kurzzeitige Schwankungen bei der Bandbreite oder Übertragung auszugleichen. Der Puffer (**590**) kann einen Playbackpuffer oder auch andere Puffer enthalten. Alternativ empfängt der Puffer (**590**) Informationen mit einer veränderlichen Rate. Vor oder nach dem Puffer (**590**) kann die komprimierte Videoinformation kanalkodiert und für eine Fehlererfassung und Berichtigung verarbeitet werden.

[0082] Der Entropiedekodierer (**580**) nimmt eine Entropiedekodierung an den entropiekodierten quantisierten Daten wie auch an der entropiekodierten Zusatzinformation (beispielsweise der Bewegungsinformation (**515**), der Größe des Quantisierungsschrittes) vor, wobei er gängigerweise die – zur Kodierung des Kodierers umgekehrte – Dekodierung vornimmt. Entropiedekodiertechniken sind unter anderem das arithmetische Dekodieren, das differentielle Dekodieren, das Huffmann-Dekodieren, das Lauflängendekodieren (run length decoding), das LZ-Dekodieren, das Wörterbuchdekodieren sowie Kombinationen hieraus. Der Entropiedekodierer (**580**) bedient sich häufig verschiedener Dekodiertechniken für verschiedene Arten von Information (beispielsweise DC-Koeffizienten, AC-Koeffizienten, verschiedene Arten von Zusatzinformation) und kann unter mehreren Kodiertabellen innerhalb einer bestimmten Dekodiertechnik auswählen.

[0083] Ist der zu rekonstruierende Rahmen (**505**) ein Vorwärtsprädiktionsrahmen, so wendet der Bewegungsausgleicher (**530**) die Bewegungsinformation (**515**) auf einen Referenzrahmen (**525**) an, um eine Prädiktion (**535**) des Rahmens (**505**) zu bilden, der gerade rekonstruiert wird. So verwendet der Bewegungsausgleicher (**530**) beispielsweise einen Makroblockbewegungsvektor, um einen Makroblock in dem Referenzrahmen (**425**) zu finden. Ein Rahmenpuffer (**520**) speichert vorher rekonstruierte Rahmen zur Verwendung als Referenzrahmen. Alternativ wendet der Bewegungsausgleicher eine andere Art des Bewegungsausgleiches an. Die Prädiktion durch den Bewegungsausgleicher ist selten vollkommen, sodass der Dekodierer (**500**) auch Prädiktionsreste rekonstruiert.

[0084] Benötigt der Dekodierer einen rekonstruierten Rahmen für einen nachfolgenden Bewegungsausgleich, so puffert der Rahmenspeicher (**520**) den rekonstruierten Rahmen zur Verwendung bei der Prädiktion des nächsten Rahmens. Bei einigen Ausführungsbeispielen wendet der Kodierer einen Deblockierfilter auf den rekonstruierten Rahmen an, um Diskontinuitäten in den Blöcken des Rahmens adaptiv zu glätten.

[0085] Ein Umkehrquantisierer (**570**) nimmt eine Umkehrquantisierung der entropiedekodierten Daten vor. Im Allgemeinen wendet der Umkehrquantisierer eine gleichmäßige skalare Umkehrquantisierung auf die entropiedekodierten Daten mit einer Schrittweite, die in Abhängigkeit von den Rahmen oder auf anderer Basis variiert, an. Alternativ wendet der Umkehrquantisierer eine andere Art von Umkehrquantisierung auf die Daten an, so beispielsweise eine nichtgleichmäßige, vektorbasierte oder nichtadaptive Quantisierung, oder er nimmt eine direkte Umkehrquantisierung der Raumdomänendaten in einem Dekodiersystem vor, bei dem keine Umkehrfrequenzwandlung verwendet wird.

[0086] Ein Umkehrfrequenzwandler (**560**) wandelt die quantisierten Frequenzdomänendaten in Raumdomänenvideoinformationen um. Bei blockbasierten Videorahmen wendet der Umkehrfrequenzwandler (**560**) eine Umkehr-DCT (inverse DCT, IDCT) oder eine Variante der IDCT auf Blöcke von DCT-Koeffizienten an, wodurch Bewegungsprädiktionsrestdaten erzeugt werden. Alternativ verwendet der Frequenzwandler (**560**) eine andere herkömmliche Umkehrfrequenzwandlung, so beispielsweise eine Fouriertransformation, oder er bedient sich einer Wavelet- oder Subbandsynthese. Bei einigen Ausführungsbeispielen wendet der Umkehrfrequenzwandler (**560**) eine Umkehrfrequenzwandlung auf Blöcke von Raumprädiktionsresten für Schlüsselrahmen an. Der Umkehrfrequenzwandler (**560**) kann eine Umkehrfrequenzwandlung der Größe 8 × 8, 8 × 4, 4 × 8 oder auch einer anderen Größe vornehmen.

III. Bewegungsabschätzung und Bewegungsausgleich

[0087] Die Interrahmenkodierung nutzt vorübergehende Redundanzen zwischen Rahmen, um eine Kompression durchzuführen. Vorübergehende Redundanzverringerungen bedienen sich vorher kodierter Rahmen als Prädiktoren bei der Kodierung des aktuellen Rahmens. Bei den nachstehend beschriebenen Ausführungsbeispielen nutzt ein Videokodierer vorübergehende Redundanzen in typischen Videosequenzen, um die Information unter Verwendung einer kleineren Anzahl von Bits zu kodieren. Der Videokodierer bedient sich der Bewegungsabschätzung, um die Bewegung eines Blocks, eines Makroblocks oder einer anderen Menge von Pixeln eines Prädiktionsrahmens relativ zu einem Referenzrahmen (beispielsweise einem vorher kodierten oder vorhergehenden Rahmen) zu parametrisieren. Der Videokodierer (wie auch der entsprechende Dekodierer) bedienen sich des Bewegungsausgleiches, um den Prädiktionsrahmen unter Verwendung der Bewegungsinformationen und des Referenzrahmens zu rekonstruieren.

[0088] Der Bewegungsausgleich ist der Vorgang der Erzeugung einer Prädiktion für einen Videorahmen (das heißt des Prädiktionsrahmens) durch Versetzen (displacing) des Referenzrahmens. Wie bereits angemerkt wurde, wird die Prädiktion für einen Block, einen Makroblock oder eine andere Menge von Daten aus dem Referenzrahmen gebildet. Darüber hinaus ist die Versetzung üblicherweise rechteckig und über die ganze Prädiktionsstrukturierung (prediction tile) konstant. Eine derartige Versetzung ist durch einen Bewegungsvektor mit zwei Komponenten entsprechend der Versetzung oder Schiebung entlang der X-Richtung und entlang der Y-Richtung festgelegt. Die horizontale Bewegungsvektorkomponente (X) und die vertikale Bewegungsvektorkomponente (Y) stellen die Versetzung zwischen der Strukturierung, die aktuell einer Prädiktion unterworfen ist, und der entsprechenden Stelle im Referenzrahmen dar. Positive Werte stellen Stellen dar, die unter der aktuellen Stelle und zu deren Rechter befindlich sind. Negative Werte stellen Stellen dar, die über der aktuellen Stelle und zu deren Linker befindlich sind.

[0089] Bei einer Implementierung ist ein Block eine 8 × 8-Strukturierung von Pixeln und ein Makroblock eine 16 × 16-Strukturierung von Pixeln, wobei die Bewegungsvektoren auf Viertelpixelgenauigkeit festgelegt sind. Bei anderen Implementierungen wenden der Kodierer und der Dekodierer eine oder mehrere der beschriebenen Techniken auf Strukturierungen verschiedener Größen oder Strukturierungen veränderlicher Größe an, und zwar mit Bewegungsvektoren verschiedener Auflösung oder beliebig veränderlichen Bewegungsvektoren und/oder unter Verwendung von Bewegungsinformationen jenseits der Bewegungsvektoren.

[0090] Die Bewegungsvektorkomponenten sind üblicherweise als Pixelversetzungen – oftmals mit Subpixelgenauigkeit – spezifiziert. Die Subpixelversetzungen werden durch Filtern des Referenzrahmens unter Verwendung geeignet festgelegter Bewegungsausgleichsfilter gefiltert. Für den Fall eines rechteckigen Subpixelbewegungsausgleiches werden die X- und Y-Komponenten als Festkommazahlen dargestellt. Der ganzzahlige Teil dieser Zahlen wird Vollpixelschiebung beziehungsweise ganzzahlige Pixelschiebung genannt, während der gebrochenzahlige Teil Subpixelschiebung genannt wird. Ist die Subpixelschiebung gleich 0, so entspricht die Bewegung einer ganzen Zahl von Pixeln. Am häufigsten ist dies als Kopie eines Blockes aus dem Referenzrahmen implementiert, um den Prädiktor zu erzeugen (obwohl gegebenenfalls bestimmte Arten der Filterung angewendet werden können). Demgegenüber wird für den Fall, dass die Subpixelschiebung nicht gleich 0 ist, der Prädiktor dadurch erzeugt, dass ein oder mehrere Filter entsprechend der Subpixelschiebung auf die

Vollpixelstellen in dem Referenzrahmen angewendet werden. Daher ist der Bewegungsausgleichsfilter durch die Subpixelschiebung bestimmt.

[0091] Zur Implementierung von Subpixelschiebungen als Filteroperationen interpolieren die Bewegungsausgleichsfilter Datenpunkte an gebrochenzahligen Pixelstellen auf Basis von Referenzwerten an Vollpixelstellen. Im Allgemeinen nimmt die Qualität der Interpolation durch Wirkung des Filters zu. Einige Ausführungsbeispiele bedienen sich trennbarer Zweihahnfilter (2-tap filter) und Vierhahnfilter (4-tap filter) in jeder Richtung, die bilinearen und bikubischen Interpolatoren entsprechen.

[0092] Bei einigen Ausführungsbeispielen bedienen sich die Bewegungsausgleichsfilter einer ganzzahligen Arithmetik und Division, die als Bitschiebung implementiert ist. Der Rundungssteuerparameter R nimmt die Werte 0 oder 1 an und bestimmt die Richtung der Rundung für diese Divisionen. Der Rundungssteuerparameter kann konstant gewählt werden, er kann durch ein externes Signal vorgegeben werden, oder er kann implizit aus vorher kodierten Informationen hergeleitet werden.

[0093] Fig. 6 zeigt Vollpixel- und Subpixelstellen in einem Referenzrahmen (**600**), der während der Subpixelbewegungsabschätzung und des Subpixelbewegungsausgleiches bei einigen Ausführungsbeispielen verwendet wird. In Viertelpixelintervallen in jeder Richtung enthält der Referenzrahmen (**600**) Subpixelstellen, für die ein Kodierer oder Dekodierer Pixelwerte für eine bestimmte Versetzung interpolieren kann. Die ganzzahligen Positionen a bis p des Referenzrahmens (**600**) sind in Fig. 6 als schattierte Kreise gezeigt, während die Viertelpixelpositionen und die Halbpixelpositionen, die zwischen den ganzzahligen Stellen beziehungsweise Vollstellen liegen, als nichtschattierte Kreise gezeigt sind. Die Stellen P_0 bis P_8 zeigen neun repräsentative Subpixelpositionen, wie in Tabelle 1 beschrieben ist.

Position	Beschreibung
P_0	Viertelpixel horizontal, Vollpixel vertikal
P_1	Halbpixel horizontal, Vollpixel vertikal
P_2	Vollpixel horizontal, Viertelpixel vertikal
P_3	Viertelpixel horizontal, Viertelpixel vertikal
P_4	Halbpixel horizontal, Viertelpixel vertikal
P_5	Vollpixel horizontal, Halbpixel vertikal
P_6	Viertelpixel horizontal, Halbpixel vertikal
P_7	Halbpixel horizontal, Halbpixel vertikal
P_8	Viertelpixel horizontal, Dreiviertelpixel vertikal

Tabelle 1: Repräsentative Subpixelpositionen

[0094] Die Dreiviertelpixelposition, als deren Beispiel P_8 angegeben ist, kann als Spezialfall der Viertelpixelposition angesehen werden; sie ist um eine Viertelpixelschiebung von einer Vollpixelstelle entfernt. Weitere Dreiviertelpixelpositionen sind möglich, jedoch nicht dargestellt. Die Subpixelpositionen P_0 bis P_8 werden im Laufe der nachfolgenden Beschreibung der Interpolationsfilter wiederverwendet. Bei alternativen Ausführungsbeispielen interpolieren der Kodierer und der Dekodierer Werte an zusätzlichen oder anderen Subpixelpositionen, so beispielsweise an jenen in einem Intervall jenseits eines Viertelpixels in jeder Richtung.

A. Genäherte bikubische Interpolationsfilter

[0095] Für Subpixelinterpolationen verwenden bei einigen Ausführungsbeispielen ein Videokodierer und ein entsprechender Dekodierer lineare/bilineare Filter und/oder kubische/bikubische Filter, die folgendermaßen festgelegt sind.

[0096] Ein linearer Interpolator ist ein lineares Polynom beziehungsweise ein Polynom erster Ordnung in einer Dimension, bei dem bekannte Werte an denjenigen zwei Gitterpunkten verwendet werden, die den zu interpolierenden Punkten am nächsten sind. Der Wert der linearen Funktion an dem interpolierten Punkt ist die lineare Interpolation. Die Multiplikatoren des linearen Polynoms werden durch Lösen eines linearen Gleichungssystems berechnet, wodurch die Koeffizienten des linearen Filters bestimmt werden. Ein linearer Interpolationsfilter wird durch zwei Filterhähne beschrieben. Ein bilinearer Interpolator ist ein linearer Interpolator, der in zwei Dimensionen trennbar ist.

[0097] Ein kubischer Interpolator ist ein kubisches Polynom beziehungsweise ein Polynom dritter Ordnung in einer Dimension, bei dem bekannte Werte an denjenigen vier Gitterpunkten verwendet werden, die dem inter-

polierten Punkt am nächsten sind. Der Wert der kubischen Funktion an dem interpolierten Punkt ist die kubische Interpolation. Die Multiplikatoren des kubischen Polynoms werden durch Lösen eines Systems von Gleichungen bestimmt, wodurch die Koeffizienten des kubischen Filters bestimmt werden. Ein kubischer Interpolationsfilter ist durch vier Filterhähne festgelegt. Ein bikubischer Interpolator ist ein kubischer Interpolator, der in zwei Dimensionen trennbar ist.

[0098] Die Begriffe „linear" und „bilinear" werden auf dem Gebiet der Videokompression und der Videodekompression üblicherweise austauschbar verwendet. Bei einer üblichen zweidimensionalen Interpolation wird die in einer Dimension ausgeführte Interpolationsoperation in der anderen Dimension repliziert, weshalb jede Filterstufe als bilineare Filterung bezeichnet wird. Die Begriffe „kubisch" und „bikubisch" sind entsprechend austauschbar.

[0099] Im Rahmen der vorliegenden Druckschrift werden die Begriffe „linear" und „bilinear" austauschbar zur Beschreibung einer Filterung in einer, zwei oder mehr Dimensionen verwendet. Auf ähnliche Weise werden die Begriffe „kubisch" und „bikubisch" austauschbar zur Beschreibung einer Filterung in einer, zwei oder mehr Dimensionen verwendet. Gleichungen (11) bis (13) legen beispielsweise Typen kubischer Filter fest, werden jedoch als bikubische Filter bezeichnet, da bei den üblichen Anwendungen der zweistufigen Interpolation für Referenzvideorahmen die Filter bei Operationen verwendet werden, die für beide Dimensionen der zweistufigen Interpolation repliziert werden. Allgemein ist die Dimensionalität der Filterung aus dem Kontext bekannt.

[0100] Bei einigen Ausführungsbeispielen verwenden ein Kodierer und ein Dekodierer genäherte bikubische Filter zur Interpolation von Werten an Subpixelstellen. Der Kodierer und der Dekodierer verwenden beispielsweise die nachfolgenden Filter (von denen F_1 ein bikubischer Filter ist, während F_2 und F_3 genäherte bikubische Filter darstellen) an möglichen Schiebungsstellen in einem Referenzrahmen, so beispielsweise bei dem in **Fig. 6** gezeigten.

Halbpixelschiebung F_1: [–1 9 9 –1] (11)

Viertelpixelschiebung F_2: [–4 53 18 –3] (12)

Dreiviertelpixelschiebung F_3: [–3 18 53 –4] (13)

[0101] In der Praxis umfassen die Filter eine Rechtsschiebung (beispielsweise 4 Bit für F_1, 6 Bit für F_2 und F_3), um die Entwicklung auszugleichen, die möglicherweise durch die Filterkoeffizienten eingeführt worden ist. Der Operator >> ist ein Rechtsschiebeoperator. Eine Rechtsschiebeoperation schiebt die Bits einer binären Zahl nach rechts, wobei das niedrigstwertige Bit gestrichen wird, und eine Null als höchstwertiges Bit hinzugefügt wird. Diese Operation führt zu einer einfachen Division durch zwei hoch der Anzahl verschobener Bits (So führt beispielsweise eine Rechtsschiebung um 3 Bit zu einer Division durch $2^3 = 8$), wobei der Rest ignoriert wird.

[0102] Die Filterkoeffizienten für F_2 und F_3 basieren unzusammenhängend auf dem echten bikubischen Viertelpixelinterpolator, der ein Vierhahnfilter ist. Die nachfolgende Gleichung zeigt das Ergebnis der Anwendung des bikubischen Viertelpixelfilters für die Stelle P_0:

(–7e + 105f + 35g – 5h) >> 7 (14)

[0103] Die Koeffizientenwerte summieren sich auf 128, und das Produkt der Filterung wird um 7 Bit nach rechts verschoben. Die genäherten bikubischen Filter F_2 und F_3 nähern sich dem reinen bikubischen Filter mit Blick auf die Leistung an, haben jedoch eine niedrigere Auflösung, was die folgende Gleichung deutlich macht.

(–7e + 105f + 35g – 5h) >> 7
= (–3,5e + 52,5f + 17,5g – 2,5h) >> 6
= (–4e + 53f + 18g – 3h) >> 6 (15)

[0104] In vielen Fällen führt die Verwendung eines reinen bikubischen Filters bei einer mehrstufigen Interpolation zu einem Verlust von Bits der Genauigkeit, sodass der Normalisierungsfaktor für den genäherten bikubischen Filter um wenigstens die Hälfte reduziert wird (das heißt, die Rechtsschiebung wird um 1 Bit oder mehr verkleinert). Die für den genäherten bikubischen Filter von Gleichung (15) gewählten Filterkoeffizienten basieren auf einer Rundung des echten bikubischen Filters, nachdem das Frequenzdomänenverhalten (beispielsweise zur Erhaltung der Hochfrequenzinformation) und das empirische Verhalten (beispielsweise zur Erhaltung der niedrigsten Verzerrung bei einer gegebenen Bitrate) berücksichtigt worden ist. Insbesondere enthal-

ten die Filter F_2 und F_3 immer noch vier Filterkoeffizienten (Im Allgemeinen beschleunigt die Verwendung von weniger Filterkoeffizienten in einem Filter die Implementierung, wobei jedoch ausreichend Filterkoeffizienten verwendet werden sollten, um das Rauschen bei einem benachbarten Pixel ausreichend zu berücksichtigen). Die Filterkoeffizienten werden derart angepasst, dass sie sich auf 64 summieren, wodurch die Implementierung unter Verwendung einer 16-Bit-Arithmetik vereinfacht wird, wobei eine Näherung eines bikubischen Filters höherer Auflösung stattfindet. Andere Filterkoeffizientenwerte, die sich auf 64 addieren, können ebenfalls verwendet werden, wobei auch hier ein bikubischer Filter angenähert wird. Ein Filter, der sich im Wesentlichen wie ein reiner bikubischer Filter verhält, der jedoch weniger Support und/oder eine niedrigere Auflösung aufweist, wird „genäherter" bikubischer Filter (approximate bicubic filter) genannt. Eine Art der objektiven Bestimmung, ob sich ein Filter im Wesentlichen wie ein reiner bikubischer Filter verhält, besteht darin nachzuprüfen, ob der genäherte Filter mit dem reinen bikubischen Filter gut (das heißt innerhalb einer vorgegebenen Schwelle) korreliert. Bei einer Implementierung wird die Korrelation durch den Kosinus des Winkels zwischen den Vektoren für die Filter (wobei gewünscht ist, dass dieser Wert so nahe wie möglich an 1 ist) gemessen, wobei die Schwelle bei 0,95 liegt. Andere objektive oder subjektive Maßnahmen, andere Messungen und/oder Schwellen können ebenfalls Verwendung finden. So können die Filterkoeffizienten des genäherten bikubischen Filters beispielsweise derart gewählt werden, dass ihre Summe einen anderen Wert ergibt, wodurch eine effiziente Fouriertransformation oder eine andere mathematische Operationen vereinfacht werden.

[0105] Wie nachstehend noch detailliert beschrieben wird, zeigt Fig. 7 allgemein die Vollpixelstellen mit Pixelwerten, die zur Berechnung interpolierter Pixel für jeden der Fälle entsprechend den bikubischen Filtern gemäß Gleichungen (11) bis (13) verwendet werden. P bezeichnet die Subpixelposition, für die ein Pixelwert berechnet wird. I_1, I_2, I_3 und I_4 stellen Vollpixelstellen entlang der Dimension der Interpolation dar. Fig. 7 zeigt die horizontale Interpolation, wobei dieselben Operationen und dieselbe Anordnung auch bei Positionen für die vertikale Interpolation Anwendung finden.

[0106] Bei alternativen Ausführungsbeispielen bedienen sich ein Kodierer und ein Dekodierer anderer und/oder zusätzlicher Interpolationsfilter. Der Kodierer und Dekodierer bedienen sich beispielsweise eines bilinearen Filters (das heißt eines Zweihahnfilters), um den Wert zu interpolieren. Mit Blick auf die Subpixelpositionen von Fig. 6 können beispielsweise diejenigen Interpolationsfilter, die zur Bestimmung der Werte P_1, P_5 und P_7 eingesetzt werden, Filter sein, die in Gleichungen (1) bis (3) dargestellt sind.

B. Eindimensionale Interpolation

[0107] Für verschiedene Subpixelpositionen berechnen der Kodierer und der Dekodierer bei einigen Ausführungsbeispielen einen interpolierten Wert in lediglich einer Dimension. Wie Fig. 7 gezeigt ist, zeigen die nachfolgenden Gleichungen den Betrieb der Filter F_1 (Halbpixelschiebung), F_2 (Viertelpixelschiebung) und F_3 (Dreiviertelpixelschiebung), wenn zwischen ganzzahligen Pixeln beziehungsweise Vollpixeln interpoliert wird.

$$F_1\ (-1I_1 + 9I_2 + 9I_3 - 1I_4 + 8 - r) >> 4 \quad (16)$$

$$F_2\ (-4I_1 + 53I_2 + 18I_3 - 3I_4 + 32 - r) >> 6 \quad (17)$$

$$F_3\ (-3I_1 + 18I_2 + 53I_3 - 4I_4 + 32 - r) >> 6 \quad (18)$$

[0108] Hierbei steuert der Wert r die Rundung. Der Wert r hängt von dem binären Rahmenniveaurundungssteuerparameter R ab und bestimmt die Interpolationsrichtung folgendermaßen.

$$r = \begin{cases} 1 - R \text{ (vertikale Richtung)} \\ R \text{ (horizontale Richtung)} \end{cases} \quad (19)$$

[0109] Zur weiteren Erläuterung einer eindimensionalen Interpolation zeigen P_1 und P_5 von Fig. 6 Halbpixelpositionen in dem Referenzrahmen (**600**), bei denen eine Interpolation in lediglich einer Dimension (das heißt die horizontale Richtung für P_1 und die vertikale Richtung für P_5) vorgenommen wird. Die folgenden Gleichungen zeigen den Betrieb des Filters F_1 (Halbpixelschiebung) bei einer Interpolation zwischen den ganzzahligen Pixeln für P_1 und P_5:

$$P_1 = (-1e + 9f + 9g - 1h + 8\ r) >> 4 \quad (20)$$

$$P_5 = (-4b + 9f + 9j - 1n + 8 - r) >> 4 \quad (21)$$

Entsprechend zeigen P_0 und P_2 von Fig. 6 Viertelpixelpositionen in dem Referenzrahmen (**600**), bei denen eine Interpolation in lediglich einer Dimension erforderlich ist. Die folgenden Gleichungen zeigen den Betrieb des Filters F_2 (Viertelpixelschiebung) bei einer Interpolation zwischen ganzzahligen Pixeln für P_0 und P_2.

$$P_0 = (-4e + 53f + 18g - 3h + 32 - r) >> 6 \quad (22)$$

$$P_2 = (-4b + 53f + 18j - 3n + 32 - r) >> 6 \quad (23)$$

[0110] Der genäherte bikubische Viertelpixelfilter F_2 kann durch lediglich eine kleine Abwandlung auch zur Berechnung von Dreiviertelpixelpositionen verwendet werden. So zeigen beispielsweise die nachfolgenden Gleichungen den Betrieb des Filters F_3 (Dreiviertelpixelschiebung) bei der Interpolation zwischen ganzzahligen Pixeln für P_8.

$$P_8 = (-3b + 18f + 53j - 4n + 32 - r) >> 6 \quad (24)$$

[0111] Alternativ bedienen sich der Kodierer und der Dekodierer anderer und/oder zusätzlicher Interpolationsfilter für um Halbpixel, Viertelpixel oder Dreiviertelpixel verschobene Positionen in einer Dimension. Der Kodierer und der Dekodierer verwenden beispielsweise Filter mit mehr oder weniger Filterkoeffizienten, anderen Filterkoeffizienten, einer anderen Rundung oder auch ohne Rundung.

C. Mehrdimensionale Interpolation

[0112] Bei einigen Ausführungsbeispielen wird die Interpolation für Subpixelpositionen vorgenommen, die in zwei Richtungen versetzt (offset) sind. In Fig. 6 beispielsweise stellen P_3, P_4, P_6 und P_7 Positionen dar, für die eine Interpolation sowohl in horizontaler wie auch in vertikaler Dimension auftritt.

[0113] Bei einem Ausführungsbeispiel, das dem Interpolationsverfahren (**800**) gemäß Fig. 8 entspricht, werden die zweidimensionalen Subpixelpositionen zunächst entlang der vertikalen Richtung und anschließend entlang der horizontalen Richtung interpoliert. Wie nachstehend noch detailliert beschrieben wird, erfolgt die Interpolation unter Verwendung eines oder mehrerer der Filter F_1, F_2 oder F_3, die in den vorstehenden Gleichungen (16) bis (18) beschrieben sind. Bei dem in Fig. 8 dargestellten Ausführungsbeispiel wird die Rundung sowohl nach der vertikalen Filterung wie auch nach der horizontalen Filterung vorgenommen. Die Bitschiebung in der Rundungsregel stellt die Beibehaltung der Genauigkeit, die innerhalb der Grenzen der 16-Bit-Arithmetik zulässig ist, in den vorläufigen Ergebnissen sicher.

[0114] In Fig. 8 wird zunächst die vertikale Filterung vorgenommen, gefolgt von einer horizontalen Filterung. Die Tatsache, dass mit der vertikalen Filterung begonnen wird, verbessert die Leistung bei einigen Architekturen. Bei anderen Ausführungsbeispielen ist die Reihenfolge der Filterungen eine andere. So kann beispielsweise die Interpolation in der horizontalen Richtung vor derjenigen in der vertikalen Richtung vorgenommen werden. Zudem sind verschiedene andere Kombinationen von Interpolationsfiltern (beispielsweise mehrere horizontale und/oder mehrere vertikale Filter und dergleichen) in Gebrauch.

[0115] Die Eingabewerte und die Ausgabepixelwerte (**811**, **838**) weisen eine Bittiefe von 8 Bit sowie einen Wertebereich mit 256 Werten auf. Die vorläufigen Werte (**820**) weisen eine Bittiefe von 16 Bit für einen Wertebereich mit 65536 Werten auf. Bei alternativen Ausführungsbeispielen weisen die Eingabewerte, die Ausgabewerte und die vorläufigen Werte eine andere (das heißt höhere) Bittiefe auf.

[0116] In einer ersten Stufe (**810**) wird ein geeigneter vertikaler Filter (F_V) auf den 8 Bit langen Eingabepixelwert beziehungsweise die entsprechenden Eingabepixelwerte (**811**) angewendet (**812**). Die Verwendung des vertikalen Filters hängt davon ab, ob die ausgewählte Subpixelposition um ein Viertelpixel, ein Halbpixel oder ein Dreiviertelpixel verschoben wird, und kann die Form eines oder mehrerer der vorstehend beschriebenen bikubischen Filter annehmen.

[0117] Die Rundungsregel nach der vertikalen Filterung ist folgendermaßen definiert.

$$(S + R_V) >> \text{shiftV} \quad (25)$$

[0118] Hierbei bezeichnet S das vertikal gefilterte Ergebnis, wobei $R_V = 2^{\text{shiftV}-1} - 1 + R$ der Rundungssteuerwert ist, der rahmenweise zwischen 0 und 1 wechselt. Die Rundungsregel enthält eine Rundung mit einer stufenweise wechselnden Rundungssteuerung (**813**) und einer Bitschiebung (**840**).

[0119] Die Rechtsschiebung bewirkt gegebenenfalls einen Auflösungsverlust, weshalb wenigstens ein Teil der Rechtsschiebung in die späteren Stufen der Interpolation verschoben wird. Der Rechtsschiebungswert für shiftV hängt von der interpolierten Subpixelposition ab. Insbesondere gilt shiftV = {5, 3, 3, 1} für P_3, P_4, P_6 und P_7. Das Ausmaß des Schiebung ist geringer, als es für einen Ausgleich der Entwicklung bedingt durch die Filterkoeffizientenwerte der ersten Stufe (das heißt die Verschiebung ist kleiner als 6 Bit für den genäherten bikubischen Filter) möglich ist, jedoch ausreichend, um sicherzustellen, dass die vorläufigen Ergebnisse der nachfolgenden Filterung im Wertebereich für die vorläufigen Werte (beispielsweise 65536 mögliche Werte für 16 Bit lange Worte) verbleiben. Verglichen mit der Vollschiebung wird durch diese verkürzte Schiebung die Genauigkeit der vorläufigen Pixelwerte (**820**) nach der ersten Stufe (**810**) der Interpolation beibehalten. Die vorläufigen Pixelwerte (**820**) weisen einen Wertebereich von y Bit auf, wobei y größer als 8 Bit ist. Das Ausmaß des Schiebung, die in der ersten Stufe durchgeführt wird, hängt gegebenenfalls von der verfügbaren Bittiefe und den Koeffizienten der Interpolationsfilter ab. So sind beispielsweise bei der hier beschriebenen beispielhaften Implementierung die vorläufigen Werte auf eine Wortgrenze von 16 Bit beschränkt.

[0120] Man betrachte den Punkt P_3 in Fig. 6 und die Eingabewerte in dem Bereich von 0 bis 255 (8 Bit). Der Bereich der vorläufigen Werte durch die Anwendung der genäherten bikubischen Filterkoeffizienten [–4 53 18 – 3) auf die 8 Bit langen Eingabewerte liegt zwischen –1785 und 18105 (etwa 14,3 Bit, wobei hier eine Rundung auf 15 Bit für die Implementierung erfolgt), was durch den Entwicklungsfaktor der Filterkoeffizienten bedingt ist. Eine nachfolgende horizontale Filterung, bei der die genäherten bikubischen Filterkoeffizienten (mit zusätzlicher Entwicklung) auf die vorläufigen Werte angewendet werden, kann Werte außerhalb des 16 Bit langen Wertebereiches erzeugen, was zu einer Bereichsüberschreitung (overflow) führt. Daher werden die vorläufigen Werte derart ausreichend verschoben, dass sichergestellt ist, dass die nachfolgende horizontale Filterung zu einem Wert innerhalb des 16 Bit langen Wertebereiches führt. So ist beispielsweise das Ausmaß der Schiebung für P_3 gleich 5 Bit, wobei der Wertebereich der verschobenen vorläufigen Werte zwischen –55 und 565 (grob 9,3 Bit, die für die Implementierung auf 10 Bit aufgerundet werden) liegt. Der Ausgabebereich durch die Anwendung der genäherten bikubischen Filterkoeffizienten auf die verschobenen vorläufigen Werte liegt dann zwischen –7860 und 40500, was einem Wertebereich von weniger als 16 Bit entspricht. Daher wird die verkürzte Schiebung derart berechnet, dass die 16 Bit lange Wortgrenze voll ausgeschöpft wird, wobei jedoch sichergestellt ist, dass in der zweiten Stufe (**830**) der Interpolation keine Überschreitung auftritt.

[0121] In der zweiten Stufe (**830**) wird ein geeigneter horizontaler Filter (F_H) eingesetzt (**832**), um den Wert an der zweidimensionalen Subpixelposition aus den Werten (**820**), die von dem vertikalen Filter bestimmt worden sind, zu interpolieren. Die Rundungsregel nach der horizontalen Filterung lautet folgendermaßen.

$$(S + 64 - R) >> 7 \qquad (26)$$

[0122] Hierbei bezeichnen S das horizontal gefilterte Ergebnis und R den Rundungssteuerwert, der rahmenweise wechselt. Wie bei der Rundungsregel der ersten Stufe enthält die Rundungsregel der zweiten Stufe eine Rundung mit stufenweise wechselnder Rundungssteuerung (**833**) und einer Bitschiebung (**834**). Aufgrund der verschobenen Schiebung in der ersten Stufe überschreitet das Ausmaß der Schiebung der zweiten Stufe üblicherweise das, was normalerweise für den ausgewählten horizontalen Filter zu erwarten ist, und wird derart errechnet, dass ein Wert ausgegeben wird, der den gewünschten Wertebereich aufweist.

[0123] Sämtliche Fälle der bikubischen Filterung können möglicherweise ein Interpolationspixel erzeugen, dessen Wert negativ ist, oder dessen Wert größer als der Maximalwert des Wertebereiches (beispielsweise 255 für eine 8 Bit lange Ausgabe) ist. In diesen Fällen schneiden bei 8 Bit langen Ausgabewerten der Kodierer und Dekodierer den Ausgabewert (**836**) derart ab, dass dieser in einem annehmbaren Bereich liegt. Insbesondere werden Bereichsunterschreitungen auf 0 und Bereichsüberschreitungen auf 255 gesetzt. Nach dem Abschneiden wird ein interpolierter 8 Bit langer Wert (**838**) ausgegeben.

[0124] In Fig. 8 liegt die Schiebung der zweiten Stufe bei 7 Bit. Es wird also ein gefilterter Ausgabewert mit 9 Bit beibehalten. Mit Blick auf das vorgenannte Beispiel für P_3 liegt beispielsweise der Bereich des gefilterten Ausgabewertes zwischen –61 und 316, was einem Wertebereich von grob 8,6 Bit entspricht (wobei für die Implementierung eine Rundung auf 9 Bit erfolgt). Obwohl der gültige Bereich der interpolierten Daten bei nur 8 Bit liegt, erzeugt das zusätzliche eine Bit im Kopfbereich (Headroom) eine Information bezüglich Bereichsüberschreitungen und Bereichsunterschreitungen. Mit anderen Worten, es tritt beim Setzen des höchstwertigen Bits (das heißt des Zeichenbits beziehungsweise des Vorzeichenbits) eine Bereichsunterschreitung oder eine Bereichsüberschreitung auf. Welche der beiden aufgetreten ist, ist insbesondere durch Analyse der verbleibenden 8 Mantissenbits ermittelbar.

[0125] Fig. 9 bis Fig. 11 zeigen die vorstehend anhand Fig. 8 beschriebene zweidimensionale Interpolation. Fig. 9 zeigt eine Subpixelposition P_7 (Halbpixel in der Horizontalen und Halbpixel in der Vertikalen) des Referenzrahmens (**600**) von Fig. 6. Zwei bikubische Halbpixelinterpolationsfilter werden zur Interpolation der Werte von P_7 verwendet. In der ersten Stufe werden die vorläufigen Werte V_1 bis V_4 aus genäherten ganzzahligen Pixelpositionen unter Verwendung eines bikubischen Halbpixelfilters der nachfolgenden allgemeinen Form berechnet.

$$V_{inter} = (-1x_1 + 9x_2 + 9x_3 - 1x_4) \quad (27)$$

[0126] Somit gilt:

$$V_1 = (-1a + 9e + 9i - 1m) \quad (28)$$

$$V_2 = (-1b + 9f + 9j - 1n) \quad (29)$$

$$V_3 = (-1c + 9g + 9k - 1o) \quad (30)$$

$$V_4 = (-1d + 9h + 91 - 1p) \quad (31)$$

[0127] Nach dem Addieren eines geeigneten Wertes von R_V werden die Ergebnisse um 1 Bit nach rechts verschoben. In der zweite Stufe werden die vorläufigen Ergebnisse V_1 bis V_4 von einem Halbpixelfilter zur Berechnung des Pixelwertes bei P_7 verwendet. Insbesondere wird ein Halbpixelfilter der nachfolgenden Form verwendet.

$$P_7 = (-1V_1 + 9V_2 + 9V_3 - 1V_4) \quad (32)$$

[0128] Wie vorstehend diskutiert wurde, wird das Ergebnis der zweiten Stufe um 7 Bit nach rechts verschoben, um einen 9 Bit langen Wert zu erhalten. Der 9 Bit lange Wert umfasst 8 Mantissenbits und ein Vorzeichenbit. Nachdem ein notwendiges Abschneiden vorgenommen worden ist, um eine Bereichsüberschreitung oder eine Bereichsunterschreitung auszugleichen, wird ein finaler 8 Bit langer interpolierter Wert ausgegeben.

[0129] Fig. 10 zeigt eine Subpixelposition P_4 (Halbpixel in der Horizontalen und Viertelpixel in der Vertikalen) des Referenzrahmens (**600**) von Fig. 6. Ein bikubischer Viertelpixel- und Halbpixelinterpolationsfilter wird zur Interpolation des Wertes P_4 verwendet. In der ersten Stufe werden die vorläufigen Werte V_1 bis V_4 aus den genäherten ganzzahligen Pixelpositionen unter Verwendung eines bikubischen Viertelpixelfilters der nachfolgenden allgemeinen Formel berechnet.

$$V_{inter} = (-4x_1 + 53x_2 + 18x_3 - 3x_4) \quad (33)$$

[0130] Der Filter wird auf dieselbe Weise auf die ganzzahligen Pixelwerte des Referenzrahmens (**600**) angewandt, wie dies vorstehend im Zusammenhang mit der Berechnung von P_7 beschrieben worden ist. Nach Addition des geeigneten Wertes von R_V werden die Ergebnisse um 3 Bit nach rechts verschoben. In der zweiten Stufe werden die vorläufigen Ergebnisse V_1 bis V_4 von einem Halbpixelfilter zur Berechnung des Pixelwertes bei P_4 verwendet. Insbesondere wird ein Halbpixelfilter der nachfolgenden Form verwendet.

$$P_4 = (-1V_1 + 9V_2 + 9V_3 - 1V_4) \quad (34)$$

[0131] Das Ergebnis der zweiten Stufe wird um 7 Bit nach rechts verschoben, um einen 9 Bit langen Wert zu erhalten. Es wird gegebenenfalls ein Abschneiden vorgenommen, und schließlich wird ein finaler 8 Bit langer interpolierter Wert ausgegeben.

[0132] Fig. 10 zeigt zudem eine Subpixelposition P_6 (Viertelpixel in der Horizontalen, Halbpixel in der Vertikalen). Um den Wert von P_6 zu interpolieren, wird die Technik zur Interpolation von P_4 mit lediglich einer geringfügigen Abwandlung angewendet. Bei dieser abgewandelten Technik wird ein bikubischer Halbpixelfilter in der ersten Stufe verwendet, um die vorläufigen Werte zu bestimmen. Die vorläufigen Pixelstellen sind in Fig. 10 bei V_5 bis V_8 gezeigt. In der zweiten Stufe verwendet der bikubische Viertelpixelfilter die vorläufigen Werte zur Berechnung der Werte von P_6. Insbesondere wird ein bikubischer Viertelpixelfilter der nachfolgenden Form verwendet.

$$P_6 = (-4V_5 + 53V_6 + 18V_7 - 3V_8) \quad (35)$$

[0133] Das Ausmaß der Schiebung in der ersten Stufe und in der zweiten Stufe sind das gleiche wie bei der Technik zur Berechnung von P_4 (das heißt, die Schiebung der ersten Stufe liegt bei 3, und die Schiebung der zweiten Stufe liegt bei 7).

[0134] Fig. 11 zeigt eine Subpixelposition P_3 (Viertelpixel in der Horizontalen, Viertelpixel in der Vertikalen) des Referenzrahmens (**600**) von Fig. 6. Zwei bikubische Viertelpixelinterpolationsfilter werden zur Interpolation des Wertes von P_3 verwendet. In der ersten Stufe werden die vorläufigen Werte V_1 bis V_4 aus den genäherten ganzzahligen Pixelpositionen unter Verwendung eines bikubischen Viertelpixelfilters der nachfolgenden allgemeinen Form berechnet.

$$V_{inter} = (-4x_1 + 53x_2 + 18x_3 - 3x_4) \quad (36)$$

[0135] Der Filter wird auf dieselbe Weise auf die ganzzahligen Pixelwerte des Referenzrahmens (**600**) angewandt, wie dies vorstehend bei der Berechnung von P_4 beschrieben worden ist. Nach der Addition des geeigneten Wertes von R_V werden die Ergebnisse um 5 Bit nach rechts verschoben. In der zweiten Stufe werden die vorläufigen Ergebnisse V_1 bis V_4 von einem weiteren bikubischen Viertelpixelfilter verwendet, um den Pixelwert bei P_3 zu berechnen. Insbesondere wird ein Viertelpixelfilter der nachfolgenden Form verwendet.

$$P_3 = (-4V_1 + 53V_2 + 18V_3 - 3V_4) \quad (37)$$

[0136] Das Ergebnis der zweiten Stufe wird um 7 Bit nach rechts verschoben, um einen 9 Bit langen Wert zu erhalten. Es wird gegebenenfalls ein Abschneiden durchgeführt, woraufhin der finale 8 Bit lange interpolierte Wert ausgegeben wird.

[0137] Obwohl in Fig. 9 bis Fig. 11 nicht gezeigt, können auch die Werte der Subpixelpositionen mit Dreiviertelsubpixelschiebungen in einer oder in beiden Dimensionen berechnet werden. Um eine derartige Subpixelposition zu berechnen, können die vorstehend dargelegten Verfahren abgewandelt werden, indem geeignete kubische Dreiviertelpixelfilter anstelle der bikubischen Viertelpixelfilter verwendet werden.

[0138] Bei anderen Ausführungsbeispielen werden bilineare Filter oder eine Kombination aus bilinearen und bikubischen Filtern zur Interpolationen der Werte an Subpixelabtastpositionen verwendet. Die Verwendung bilinearer Filter verringert im Allgemeinen das Ausmaß der Schiebung, die (nach der ersten Stufe und insgesamt) vorgenommen wird, da die Koeffizienten eine kleinere Entwicklung als bei bikubischen Filtern einbringen. Bei einer Implementierung unter Verwendung bilinearer Filter und 16 Bit langer vorläufiger Werte wird beispielsweise keine Bitschiebung in der ersten Stufe ausgeführt, wodurch die Verwendung der 16 Bit langen Wortgrenze maximiert wird, und die Rechtsschiebung von 4 Bit nach der letzten Stufe ausgeführt wird. Ähnlich kann das Abschneiden bis in die letzte Stufe verschoben werden.

[0139] Eines der den beschriebenen Verfahren zugrundeliegenden Prinzipien betrifft die Verwendung der höchstmöglichen Genauigkeit in jeder Filterstufe, während man innerhalb einer gewünschten „Wortgrößengrenze" W verbleibt. Weist der Ausgabewert einen Wertebereich von D Bit auf, und werden L Bit in der letzten Stufe ausgesondert, so kann die Ausgabe der letzten Filterstufe bis zu D + L + 1 Bit annehmen, wobei das eine zusätzliche Bit zur Signalisierung von Bereichsunterschreitungen und Bereichsüberschreitungen verwendet wird. Andersherum gesagt, wenn die letzte Stufe der Filterung zu einer Entwicklung von k Bit führt, so sollte die Eingabe für die letzte Stufe innerhalb von D + L – k liegen. Um die maximale Genauigkeit bei einer W Bit langen Darstellung zu erhalten, ist die nachfolgende Beziehung gegeben.

$$D + L + 1 = W \quad (38)$$

[0140] Darüber hinaus ist die Eingabe für die letzte Stufe D + L – k = W – k – 1 Bit lang.

[0141] Die vorgenannte Logik kann rekursiv auf die vorletzte Stufe der Filterung und so weiter angewendet werden. In der Tat können die Schranken zusammengeschoben werden, indem gebrochenzahlige Bits zur Darstellung von nicht-2^k-Bereichen und Entwicklungsfaktoren verwendet werden.

[0142] Fig. 12 bis Fig. 15 sind Diagramme, die verschiedene Techniken darstellen, die vorstehend in Kombination beschrieben wurden, die jedoch auch einzeln auf eine mehrstufige Interpolation anwendbar sind. Fig. 12 bis Fig. 15 zeigen nicht die verschiedenen Arten, auf die die jeweilige mehrstufige Interpolation (**1200**, **1300**, **1400**, **1500**) in Verbindung mit den anderen mehrstufigen Interpolationstechniken verwendet werden kann.

[0143] Obwohl jede der Fig. 12 bis Fig. 15 zwei Stufen darstellt, kann auch die Technik der mehrstufigen Interpolation (**1200**, **1300**, **1400**, **1500**) gemäß Darstellung in Fig. 12 bis Fig. 15 mehr Stufen umfassen. Allgemeiner gesagt, es können Techniken der mehrstufigen Interpolation (**1200**, **1300**, **1400**, **1500**) mit einem beliebigen Typ trennbarer Filter in mehreren Dimensionen wie auch als beliebige Filter implementiert sein, die in einer Kaskaden-, Gitter- oder Netzstruktur implementiert sind.

[0144] Fig. 12 bis Fig. 15 zeigen verallgemeinerte Eingabewerte, Ausgabewerte und Filter, die bei der mehrstufigen Interpolation Verwendung finden. Die jeweils spezifische Wahl der Bittiefe für die Eingabewerte der ersten Stufe, die Ausgabewerte der letzten Stufe und die vorläufigen Werte kann willkürlich in Entsprechung zu den technischen Spezifikationen einer Zielarchitektur oder Zielanwendung erweitert werden. So können beispielsweise die Eingabewerte 8 Bit lange Pixelwerte an Vollpixelpositionen im Referenzrahmen, die Ausgabewerte 8 Bit lange Pixelwerte an Subpixelpositionen im Referenzrahmen und die Filter reguläre und genäherte bikubische Filter (wie vorstehend anhand Fig. 6 bis Fig. 8 beschrieben) sein. Alternativ weisen die Eingabewerte und/oder die Ausgabewerte Wertebereiche mit einer anderen Bittiefe auf, oder es können andere Filter eingesetzt werden.

[0145] Eine Komponente, so beispielsweise der Kodierer oder Dekodierer, die vorstehend anhand Fig. 4 beziehungsweise 5 beschrieben worden ist, kann die mehrstufige Interpolation (**1200**, **1300**, **1400**, **1500**) ausführen. Alternativ kann ein anderer Kodierer oder Dekodierer oder eine andere Art von Komponente die mehrstufige Interpolation (**1200**, **1300**, **1400**, **1500**) ausführen.

[0146] Fig. 12 zeigt ein Diagramm einer mehrstufigen Interpolation (**1200**) mit einem vergrößerten Wertebereich (in Bit) für vorläufige interpolierte Werte. In der ersten Stufe (**1210**) wendet die Komponente einen ersten Filter F_1 auf einen oder mehrere Eingabewerte (**1211**) in einem Bereich von x Bit an (**1212**), wodurch ein vorläufiger Wert oder mehrere vorläufige Werte (**1220**) in einem Bereich von y Bit erzeugt werden, wobei y größer als x ist. Die y Bit langen vorläufigen Werte sind beispielsweise Pixelwerte mit einem Wertebereich von mehr als 8 Bit, und die x Bit langen Eingabewerte weisen einen Wertebereich von 8 Bit auf.

[0147] Bei jeder der nicht vorhandenen oder mehreren vorläufigen Stufen (**1222**) wendet, was nicht im Detail gezeigt ist, die Komponente einen Filter auf die vorläufigen Werte (**1220**) in einem Bereich von y Bit an. Die Ausgabe der vorläufigen Stufen ist ein vorläufiger Wert (**1229**) in einem Bereich von z Bit, wobei z größer als x ist (In Fig. 12 bis Fig. 15 sind für den Fall, dass die letzte Stufe die zweite Stufe ist, die von der ersten Stufe ausgegebenen vorläufigen Werte die vorläufigen Eingabewerte für die letzte Stufe).

[0148] In der letzten Stufe (**1230**) wendet die Komponente einen letzten Filter F_L auf die vorläufigen Werte (**1229**) in einem Bereich von z Bit an (**1232**). Die finale Ausgabe ist ein Ausgabewert (**1234**) in einem Bereich von x Bit. Für jede der Techniken der mehrstufigen Interpolation (**1200**, **1300**, **1400**, **1500**) wiederholt die Komponente gegebenenfalls die mehrstufige Interpolation (**1200**, **1300**, **1400**, **1500**) für zusätzliche Ausgabewerte. Bei der wiederholten Interpolation kann die Komponente bestimmte vorläufige Werte, die bei den vorhergehenden Interpolationen berechnet worden sind, erneut verwenden.

[0149] Fig. 13 zeigt ein Diagramm einer mehrstufigen Interpolation (**1300**) mit ausgelassenem Abschneiden. Das Verschieben des Abschneidens beschleunigt die Berechnung beispielsweise dadurch, dass die Komponente nicht mehr jeden vorläufigen Wert hinsichtlich einer oberen und unteren Schranke des Bereiches prüft. Das Verschieben des Abschneidens wahrt auch die Genauigkeit bei den vorläufigen Werten.

[0150] In der ersten Stufe (**1310**) wendet die Komponente einen ersten Filter F_1 auf einen oder mehrere Eingabewerte (**1311**) in einem Bereich von x Bit an (**1312**). Nach Anwenden des ersten Filters F_1 wird kein Abschneiden vorgenommen. So können der eine vorläufige Wert oder die mehreren vorläufigen Werte (**1320**), die von dem ersten Filter F_1 ausgegeben werden, einen Wertebereich von mehr als x Bit aufweisen. Die Eingabewerte sind beispielsweise 8 Bit lange Werte, und die Ausgabe des ersten Filters F_1 weist aufgrund des Entwicklungsfaktors, der durch die Koeffizienten des ersten Filters F_1 eingeführt worden ist, einen Wertebereich von 9 oder mehr Bit auf.

[0151] Bei jeder der nicht vorhandenen oder mehreren vorläufigen Stufen (**1322**) wendet, was nicht im Detail gezeigt ist, die Komponente einen Filter auf den einen oder die mehreren nicht abgeschnittenen vorläufigen Werte (**1320**) an. Das Abschneiden kann auch in den nicht vorhandenen oder mehreren vorläufigen Stufen (**1322**) ausgelassen werden. Die von den nicht vorhandenen oder mehreren vorläufigen Stufen (**1322**) ausgegebenen vorläufigen Werte (**1329**) werden in die letzte Stufe (**1330**) eingegeben, in der die Komponente einen letzten Filter F_L auf die Werte (**1329**) anwendet (**1332**). Die finale Ausgabe aus dem letzten Filter F_L wird ab-

geschnitten (**1334**), und ein Wert (**1336**) in einem Bereich von x Bit wird ausgegeben.

[0152] Fig. 14 zeigt ein Diagramm einer mehrstufigen Interpolation (**1400**) mit verschobener Bitschiebung. In der ersten Stufe (**1410**) wendet die Komponente einen ersten Filter F_1 auf einen oder mehrere Eingabewerte (**1411**) in einem Bereich von x Bit an (**1412**). Im Zusammenwirken mit der Anwendung des ersten Filters F_1 oder auch nach dessen Anwendung wird eine verkürzte Bitschiebung (**1414**) ausgeführt. Die verkürzte Schiebung (**1414**) ist kleiner als diejenige, die benötigt wird, um einen Ausgabewert (angesichts des Entwicklungsfaktors der Koeffizienten des ersten Filters F_1) in einem Bereich von x Bit sicherzustellen, und ist kleiner als diejenige, die üblicherweise mit dem ersten Filter F_1 assoziiert wird. Entsprechend erzeugt die verkürzte Bitschiebung (**1414**) einen oder mehrere vorläufige Werte mit einem Wertebereich (von y Bit), der größer als x Bit ist. So weisen die Eingabewerte beispielsweise einen Wertebereich von 8 Bit auf, und die vorläufigen Werte einen Wertebereich von mehr als 8 Bit.

[0153] In jeder der nicht vorhandenen oder mehreren vorläufigen Stufen (**1422**) wendet, was im Detail nicht gezeigt ist, die Komponente einen Filter auf einen oder mehrere vorläufige Werte (**1420**) an. Ein oder mehrere vorläufige Werte (**1429**) mit einem Wertebereich von z Bit (größer als x Bit) werden aus den nicht vorhandenen oder mehreren vorläufigen Stufen (**1422**) ausgegeben, und in einer letzten Stufe (**1430**) wenden die Komponenten den letzten Filter F_L auf die Werte (**1429**) an (**1432**). Die finale Ausgabe aus dem letzten Filter F_L wird in einem Ausmaß verschoben (**1434**), das größer als das üblicherweise mit dem letzten Filter F_L assoziierte Ausmaß ist, wodurch der Wertebereich des Ausgabewertes (**1434**) auf eine spezifizierte Bittiefe kontrahiert wird. So ist beispielsweise der Wertebereich (in Bit) des Ausgabewertes (**1434**) gleich x oder x + 1. Bei einer Implementierung wird die Schiebung der ersten Stufe und der vorläufigen Stufe soweit als möglich bis zur finalen Stufe verschoben. Das Ausmaß, in dem die Schiebung verschoben wird, kann von der verfügbaren Bittiefe für die vorläufigen Berechnungen und den Entwicklungsfaktoren der jeweiligen Filter abhängen.

[0154] Fig. 15 zeigt eine Technik der mehrstufigen Interpolation (**1500**) und der Verwendung einer stufenweise wechselnden Rundungssteuerung. Die mehreren Stufen in der Interpolationstechnik (**1500**) wechseln dahingehend, wie sie die Rundungssteuerung zur Anpassung der Rundung einsetzen. Dies trägt dazu bei, bei bestimmten Videosequenzen eine Häufung von Rundungsfehlern von einem Rahmen zum nächsten zu verhindern. Enthält beispielsweise eine Videosequenz niedriger Qualität eine allmähliche Bewegung in einer Dimension (panning) oder zwei Dimensionen (zooming), so kann die Häufung von Rundungsfehlern zu einem allmählichen von Rahmen zu Rahmen stärker werdenden Verblassen der Farben führen, was wahrnehmbare Artefakte bewirkt. Ein stufenweise erfolgender Wechsel der Rundungssteuerung trägt dazu bei, dass dieses Farbverblassen (color fading) verhindert wird.

[0155] Ein numerisches Beispiel soll helfen, die Rundung zu erläutern, bei der die stufenweise wechselnde Rundungssteuerung vor einer Rechtsbitschiebung zum Einsatz kommt. Die Rechtsbitschiebung führt im Wesentlichen zur Division und Trunkierung des nach rechts geschobenen Wertes. Das Addieren eines Rundungswertes vor der Bitschiebung bewirkt eine Rundung des geschobenen Wertes nach oben oder nach unten (zur jeweils nächsten ganzen Zahl), anstatt dass immer eine Rundung nach unten (Trunkierung) erfolgen würde. Der Einsatz der Rundungssteuerung ändert bei einem Marginalwert die Richtung der Rundung (nach oben oder nach unten). Es sei beispielsweise für jede der mehreren Stufen angenommen, dass die Ausgabe der Filterung durch Addieren von ½ des „Divisors" einer Rechtsschiebung vor der Rechtsschiebung angepasst wird (beispielsweise das Addieren von $2^4 = 16$ vor einer 5-Bit-Rechtsschiebung oder das Addieren von $2^6 = 64$ vor einer 7-Bit-Rechtsschiebung). Die Wirkung der Addition besteht in der (auf die nächste höhere ganze Zahl erfolgenden) Aufrundung von Werten, die eine gebrochenzahlige Komponente von 0,5 oder höher (nach Division entsprechend der Bitschiebung) aufweisen. Derartige Werte würden andernfalls durch die Rechtsschiebung (auf die nächste niedrigere ganze Zahl) trunkiert werden. Unabhängig von der Addition werden Werte, die eine gebrochenzahlige Komponente von weniger als 0,5 (nach Division entsprechend der Bitschiebung) aufweisen, immer noch durch die Rechtsschiebung (auf die nächste niedrigere ganze Zahl) trunkiert. Die Rundungssteuerung wechselt dann die Richtung der Rundung für bestimmte Marginalwerte. So wird beispielsweise in jeder der mehreren Stufen die Ausgabe der Filterung durch Subtrahieren von 0 oder 1 (des wechselnden Rundungssteuerwertes) vor der Rechtsschiebung (das heißt $2^{shiftV-1}$ oder $2^{shiftV-1} - 1$) angepasst. Der Effekt der Rundungssteueranpassung besteht darin, die Richtung der Rundung für Werte zu ändern, die eine gebrochenzahlige Komponente von 0,5 (nach Division entsprechend der Bitschiebung) aufweisen. Wird 1 subtrahiert, so werden die Marginalwerte abgerundet. Andernfalls werden die Marginalwerte aufgerundet.

[0156] Jede der mehreren Stufen setzt einen Rundungssteuerwert ein, der vor der mehrstufigen Interpolation zwischen 0 und 1 wechselt, sodass die verschiedenen Stufen einen Wechsel dahingehend vornehmen, wie die Rundungssteuerwerte eingesetzt werden. Alternativ verwendet die mehrstufige Interpolationstechnik (**1500**)

einen Rundungssteuerwert, der selbst stufenweise wechselt.

[0157] In der ersten Stufe (**1510**) von Fig. 15 wendet die Komponente einen ersten Filter F_1 auf einen oder mehrere Eingabewerte (**1511**) in einem Bereich von x Bit an (**1512**). Im Zusammenwirken mit der Anwendung des ersten Filters F_1 oder nach dessen Anwendung wird eine Rundung an der Ausgabe des ersten Filters F_1 vorgenommen (**1514**). Die Rundung (**1514**) wird mittels der stufenweise wechselnden Rundungssteuerung angepasst. So bewirkt in der ersten Stufe (**1510**) die stufenweise wechselnde Rundungssteuerung beispielsweise eine Rundung des Ausgabewertes nach oben zur nächsten ganzen Zahl, wenn der Ausgabewert ein Marginalwert ist (wobei der Ausgabewert andernfalls nach unten gerundet werden wird). Ein oder mehrere gerundete vorläufige Werte (**1520**) werden von der ersten Stufe in die zweite Stufe (**1530**) eingegeben.

[0158] In der zweiten Stufe (**1530**) wendet die Komponente einen zweiten Filter F_2 auf einen oder mehrere vorläufige Werte (**1520**) an (**1532**). Die Rundung (**1534**) wird an der Ausgabe des zweiten Filters F_2 vorgenommen. Im Zusammenwirken mit der Anwendung des zweiten Filters F_2 oder nach dessen Anwendung wird eine Rundung (**1534**) mit stufenweise wechselnder Rundungssteuerung vorgenommen, wobei die Rundungssteuerung die Rundung in entgegengesetzte Richtung, wie dies bei den Marginalwerten der erste Stufe der Fall war, vornimmt. So bewirkt beispielsweise in der zweiten Stufe (**1530**) die stufenweise wechselnde Rundungssteuerung eine Rundung des Ausgabewertes nach unten zur nächsten ganzen Zahl, wenn der Ausgabewert der Marginalwert ist. Ein oder mehrere vorläufige Werte (**1536**) werden von der zweiten Stufe ausgegeben und können in nicht vorhandenen oder mehreren zusätzlichen Stufen (**1540**) verwendet werden. Die nicht vorhandenen oder mehreren zusätzlichen Stufen (**1540**) können darüber hinaus eine stufenweise wechselnde Rundungssteuerung enthalten.

[0159] Die wechselnde Rundungssteuerung ist nicht auf eine Anwendung in aufeinanderfolgenden Stufen beschränkt, sondern kann auch bei verschiedenen anderen Kombinationen der Stufen Verwendung finden. Die erste Richtung kann darüber hinaus von einer Anzahl von Parametern abhängig sein. So kann die erste Richtung beispielsweise bei einem Videokodierer oder einem entsprechenden Dekodierer von der Rundungssteuerung, die bei den vorherigen Rahmen verwendet worden ist, oder vom Typ des Rahmens, der interpoliert wird (beispielsweise ein I-Rahmen, ein P-Rahmen oder ein B-Rahmen) abhängen. Bei anderen Ausführungsbeispielen kann die erste Richtung konstant gewählt werden, sie kann implizit aus Kausalinformation (beispielsweise aus vorher kodierter/dekodierter Information) hergeleitet werden, sie kann unter Verwendung eines Pseudozufallsgenerators hergeleitet werden, oder sie kann als Teil des Bitstroms signalisiert werden. Die stufenweise wechselnde Rundungssteuerung kann bei einer mehrstufigen Interpolation unter Verwendung eines beliebigen Filters aus einer Vielzahl von Interpolationsfiltern, darunter bilineare Filter, bikubische Filter oder genäherte bikubische Filter, Verwendung finden.

D. Chrominanzbewegungsvektoren

[0160] Da Chrominanzbewegungsvektoren (Chroma-Bewegungsvektoren) implizit aus den kolokalisierten Luminanzbewegungsvektoren abgeleitet werden, ist ihre Genauigkeit beschränkt und bietet Raum für Vereinfachung. Die Vereinfachung kann die Berechnungskomplexität der Subpixelinterpolation für Chrominanzwerte bei dem Kodierer und dem Dekodieret verringern, ohne dass die Wahrnehmungsqualität des kodierten Videos merklich sinkt. Darüber hinaus können der Kodierer und der Dekodierer zwischen verschiedenen Betriebsarten (Modi) für die Chrominanzbewegungsvektorrundung und die zugehörige Interpolation hin- und herschalten. Eine Betriebsart beispielsweise legt Wert auf die Qualität des kodierten Videos auf Kosten einer größeren Berechnungskomplexität. Eine weitere Betriebsart betont die Einfachheit der Berechnung auf Kosten der Qualität.

[0161] Bei einer Implementierung verwenden ein Videokodierer und der entsprechende Dekodierer ein 1-Bit-Sequenzlevel-Feld „FASTUVMC" zur Steuerung der Subpixelinterpolation für Chrominanzwerte und der Rundung der Chrominanzbewegungsvektoren. Aus diesem Grunde operieren der Videokodierer und der entsprechende Dekodierer in einem von zwei verschiedenen Chrominanzrundungsbetriebsarten, nämlich einem Schnellmodus und einem Grundmodus, getrennt.

[0162] Fig. 16 zeigt eine Technik (**1600**) zum Auswählen zwischen mehreren Chrominanzrundungs- und Interpolationsbetriebsarten beziehungsweise Modi. Ein Videokodierer oder ein entsprechender Dekodierer, so beispielsweise gemäß der Beschreibung von Fig. 4 beziehungsweise 5, kann diese Techniken ausführen.

[0163] Der Videokodierer oder Dekodierer bestimmt (**1610**), ob der 1 Bit lange Merker FA-STUVMC die Verwendung des Schnellmodus für den Chrominanzbewegungsausgleich (flag = 1) oder des Grundmodus für den Chrominanzbewegungsausgleich (flag – 0) anzeigt. Der Merker ist beispielsweise ein Sequenzlevelfeld ent-

sprechend einer Einstellung des Anwenders, die der Kodierer in den Bitstrom des kodierten Videos hineinschreibt, und der Dekodierer aus dem Bitstrom ausliest. Alternativ bedienen sich der Kodierer und der Dekodierer mehrerer Bits zur Signalisierung einer Chrominanzrundung und/oder einer Interpolationsbetriebsart unter Verwendung von Kodes fester Länge oder veränderlicher Länge, beispielsweise um zwischen mehr als zwei verfügbaren Betriebsarten auszuwählen. Anstelle eines Sequenzlevelfeldes entsprechend einer Einstellung seitens eines Anwenders kann die Schaltinformation auch anderswo in dem Bitstrom signalisiert und/oder entsprechend anderen Kriterien eingestellt werden.

[0164] Der Videokodierer oder der entsprechende Dekodierer nehmen anschließend einen Chrominanzbewegungsausgleich im Grundmodus (**1620**) oder im Schnellmodus (**1630**) vor. Die Details betreffend den Grundmodus (**1620**) und den Schnellmodus (**1630**) bei der Bewegungsvektorrundung und der zugehörigen Interpolation sind nachstehend für eine Implementierung angegeben. Alternativ weisen diese Modi andere Implementierungen auf. So kann beispielsweise die Nachschlagtabelle, die bei der Implementierung mit Schnellmodus (**1630**) gemäß nachstehender Beschreibung verwendet wird, zu einer anderen Abbildung geändert werden, was zu einem gewünschten Leistungsniveau für eine spezifische Architektur führt, oder sie kann geändert werden, um Bewegungsvektoren anderer Genauigkeit zu entsprechen. Zusätzlich zum Grundmodus (**1620**) und zum Schnellmodus (**1630**) oder anstelle hiervon können sich der Kodierer oder der Dekodierer anderer Modi für die Chrominanzbewegungsvektorrundung der Chrominanzbewegungsvektoren bedienen.

[0165] Bei einer Implementierung werden im Schnellmodus (beispielsweise wenn der Chrominanzrundungsmerker gleich 1 ist) die Chrominanzbewegungsvektoren in Viertelpixelversetzungen (das heißt in Einviertelpixel- und Dreiviertelpixelversetzungen) auf die nächsten Vollpixelpositionen abgerundet, die Chrominanzbewegungsvektoren, die an Halbpixelversetzungen befindlich sind, werden ungerundet gelassen, und die bilineare Filterung wird für die gesamte Chrominanzinterpolation vorgenommen. In diesem Modus steigt die Geschwindigkeit beim Kodierer und beim Dekodierer. Die Motivation für diese Optimierung ist der merkliche Unterschied zwischen der Komplexität der Interpolationspixelversetzungen, die bei (a) Vollpixelpositionen, (b) Halbpixelpositionen, (c) Viertelpixelpositionen für wenigstens eine Koordinate (x und y) und (d) Viertelpixelpositionen für beide Koordinaten auftreten. Das Verhältnis a : b : c : d ist grob 1 : 4 : 4,7 : 6,6. Durch Anwenden des Schnellmodus kann (a) und (b) der Vorzug gegeben werden, was die Dekodierzeit senkt. Da dies nur mit Blick auf die Chrominanzinterpolation erfolgt, sind die Qualitätsverluste bei Kodierung und Qualität (insbesondere die sichtbare Qualität) vernachlässigbar.

[0166] Im Schnellmodus wird ein finales Rundungslevel bei den Chrominanzbewegungsvektoren folgendermaßen vorgenommen:

```
// RndTbl[–3] = –1, RndTbl[–2] = 0, RndTbl[–1] = +1, RndTbl[0] = 0
// RndTbl[1] = –1, RndTbl[2] = 0, RndTbl[3] = 1
cmv_x = cmv_x + RndTbl[cmv_x % 4];
cmv_y = cmv_y + RndTbl[cmv_y % 4];                                    (39)
```

[0167] Hierbei bezeichnen cmv_x und cmv_y die x- und y-Koordinaten der Chrominanzbewegungsvektoren in Einheiten von Viertelpixeln, wobei % die Modulooperation (Restoperation) bezeichnet, für die (x % a) = –(–x % a) gilt (Der Modulowert einer negativen Zahl ist gleich dem negativen Modulowert der entsprechenden positiven Zahl). Ist also cmv_x (oder cmv_y) durch 4 teilbar, so weist der Chrominanzbewegungsvektor eine ganzzahlige Versetzung auf. Ist cmv_x % 4 = +/–2, so weist der Chrominanzvektor eine Halbpixelversetzung auf. Ist cmv_x % 4 = +/–1 oder +/–3, so weist der Chrominanzvektor eine Viertelpixelversetzung auf. Wie aus der vorgenannten Abbildungsoperation ersichtlich ist, sind die Viertelpixelpositionen durch Rundung des Chrominanzbewegungsvektors auf die nächste Vollposition beziehungsweise ganzzahlige Position (Halbpixelpositionen bleiben unverändert) nicht zugelassen. Dieser Modus führt also eine Abbildung der Chrominanzkoordinate auf Vollpixel- und Halbpixelpositionen durch. Eine bilineare Filterung kann zum Zwecke einer weiteren Beschleunigung für sämtliche Chrominanzinterpolationen in diesem Modus vorgenommen werden. Obwohl diese Schnellmodusimplementierung in Kombination mit der Auswahl zwischen mehreren Rundungsmodi beschrieben worden ist, kann die Schnellmodusimplementierung alternativ auch unabhängig verwendet werden (das heißt als einzig möglicher Modus).

[0168] Fig. 17 ist eine Tabelle (**1700**), die den Schnellmodus der Chrominanzrundung zeigt. Die erste Reihe (**1710**) zeigt die Luminanzbewegungsvektorwerte mit Viertelpixelgenauigkeit. Obwohl die Luminanzbewegungsvektorwerte bezüglich ihrer gebrochenzahligen Versetzungen von den Vollpixelwerten gezeigt sind, können sie als ganzzahlige Werte dargestellt werden, wobei jede ganze Zahl ein Viertelpixelinkrement (das heißt 0, 1, 2, 3, 4 anstelle von 0, ¼, ½, ¾, 1) darstellt. Die zweite Reihe (**1720**) zeigt, wie die entsprechenden Chro-

minanzbewegungsvektorwerte im Schnellmodus derart gerundet werden, dass sie Vollpixel- oder Halbpixelgenauigkeit aufweisen.

[0169] In dem zweiten Modus, das heißt dem Grundmodus dieser Implementierung (wenn beispielsweise der Chrominanzrundungsmerker gleich 0 ist), bleiben die Chrominanzbewegungsvektoren, die an Viertelpixelversetzungen befindlich sind, ohne Rundungsfehler an ihren Viertelpixelversetzungen. Die Chrominanzbewegungsvektoren an anderen Subpixelversetzungen werden auf die nächste Vollpixel- oder Halbpixelposition gerundet. In diesem Modus ist die Geschwindigkeit des Dekodierens geringer als in den anderen Modi, wobei jedoch die Genauigkeit, mit der die Chrominanzpixelwerte berechnet werden, höher ist. Der Grundmodus nimmt daher eine Abbildung der Chrominanzkoordinaten auf ganzzahlige Halbpixel- und Viertelpixelpositionen vor. Eine bikubische oder bilineare Filterung gemäß vorstehender Beschreibung kann zur Chrominanzinterpolation verwendet werden.

[0170] Fig. 18 ist eine Tabelle, die den Grundmodus der Chrominanzrundung darstellt. Die erste Reihe (**1810**) zeigt die Luminanzbewegungsvektorwerte mit Viertelpixelgenauigkeit. Die dritte Reihe (**1830**) zeigt, wie die entsprechenden Chrominanzbewegungsvektorwerte im Grundmodus gemäß vorstehender Beschreibung derart abgerundet werden, dass sie Vollpixel-, Halbpixel- oder Viertelpixelgenauigkeit aufweisen. Bei anderen Implementierungen wird der Chrominanzraum auf andere Auflösungen im Chrominanzraum gerundet.

[0171] Eingedenk der vorbeschriebenen dargelegten Prinzipien der vorliegenden Erfindung unter Bezugnahme auf die Ausführungsbeispiele ist einsichtig, dass die verschiedenartigen Ausführungsbeispiele mit Blick auf Anordnung und Details modifiziert werden können, ohne dass von den zugrundeliegenden Prinzipien abgewichen würde. So sind die Prinzipien und Techniken gemäß vorstehender Beschreibung beispielsweise nicht auf die Verwendung in einem Videokodierer und/oder Videodekodierer beschränkt. Anstatt dessen können sie in einem beliebigen Berechnungskontext eingesetzt werden, wo ein Wert teilweise auf Basis eines oder mehrerer vorläufiger Werte berechnet wird, oder wo trennbare Filter in mehreren Dimensionen eingesetzt werden.

[0172] Es ist einsichtig, dass die Programme, Vorgänge und Verfahren gemäß vorstehender Beschreibung nicht auf eine bestimmte Art von Berechnungsumgebung bezogen oder beschränkt sind, es sei denn, es wird explizit anderes angegeben. Bestimmte Arten von Allzweck- oder Spezialzweckberechnungsumgebungen können zur Ausführung der Operationen entsprechend der vorstehend beschriebenen Techniken verwendet werden. Elemente der gezeigten Ausführungsbeispiele, die als Software dargestellt sind, können als Hardware implementiert sein, und umgekehrt.

[0173] Eingedenk der zahlreichen möglichen Ausführungsbeispiele, bei denen die Prinzipien der vorliegenden Erfindung Anwendung finden können, werden für die vorliegende Erfindung all diejenigen Ausführungsbeispiele beansprucht, die in den Schutzumfang der beiliegenden Ansprüche und zugehörigen Äquivalente fallen.

Patentansprüche

1. Videokodier- oder -dekodierverfahren, das in einem Computersystem (**300**) implementiert ist und umfasst:
Auswählen (**1610**) eines von mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren;
Berechnen („computing") eines Leuchtdichtebewegungsvektors (Helligkeitsbewegungsvektors; „luminance motion vector") für einen Satz von Pixelwerten in einem vorhergesagten Videorahmen bezüglich eines Referenzvideorahmens im Leuchtdichteraum; und
Konvertieren des Leuchtdichtebewegungsvektors in einen Chrominanzbewegungsvektor, wobei die Konversion in einem von mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren durchgeführt wird, wobei im Vergleich zu einem ersten Modus (**1620**) der mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren ein zweiter Modus (**1630**) der mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren die Präzision des Chrominanzbewegungsvektors verringert und dadurch die Berechnungskomplexität („computational complexity") nachfolgender Bewegungskompensation reduziert.

2. Verfahren nach Anspruch 1, wobei die mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren durch einen Benutzer für die Benutzung bei der Bewegungskompensation auswählbar sind.

3. Verfahren nach Anspruch 1 oder 2, wobei die mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren vor der Kodierung auswählbar sind.

4. Verfahren nach einem der Ansprüche 1 bis 3, wobei ein einzelner der mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren für eine Videosequenz ausgewählt wird.

5. Verfahren nach einem der Ansprüche 1 bis 4, wobei gemäß des ersten Modus' (**1620**) die Konversion ein Runden des Chrominanzbewegungsvektors auf Viertelpixel-Genauigkeit im Chrominanzraum beinhaltet.

6. Verfahren nach Anspruch 5, wobei gemäß des zweiten Modus' (**1630**) die Konversion ein Runden des Chrominanzbewegungsvektors auf Halbpixel-Genauigkeit im Chrominanzraum beinhaltet.

7. Verfahren nach einem der Ansprüche 1 bis 6, wobei der Leuchtdichtebewegungsvektor einen Viertelpixelwert in einer Dimension im Leuchtdichteraum hat und wobei der Chrominanzbewegungsvektor einen entsprechenden Ganzpixel- oder Halbpixelwert in der einen Dimension im Chrominanzraum hat.

8. Verfahren nach einem der Ansprüche 1 bis 6, wobei der Leuchtdichtewegungsvektor einen Halbpixelwert in einer Dimension im Leuchtdichteraum hat und wobei der Chrominanzbewegungsvektor einen entsprechenden Viertelpixelwert in der einen Dimension im Chrominanzraum hat.

9. Verfahren nach einem der Ansprüche 1 bis 8, wobei ein Signal in einem Videobitstrom einen der mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren angibt.

10. Verfahren nach Anspruch 9, wobei das Signal ein Ein-Bit-Wert in einem Sequenzlevelfeld des Videobitstroms ist.

11. Verfahren nach einem der Ansprüche 1 bis 4, wobei die Berechnung des Leuchtdichtebewegungsvektors mit Viertelpixel-Genauigkeit im Leuchtdichteraum durchgeführt wird; und wobei die Konversion des Leuchtdichtebewegungsvektors in den Chrominanzbewegungsvektor mit Viertelpixel-Genauigkeit im Chrominanzraum durchgeführt wird.

12. Verfahren nach Anspruch 11, wobei die Chrominanzbewegungsvektorskala eine halbe Leuchtdichtebewegungsvektorskala ist.

13. Verfahren nach Anspruch 11 oder 12, wobei der Chrominanzbewegungsvektor einen Viertelpixelwert in wenigstens einer Dimension im Chrominanzraum hat.

14. Verfahren nach Anspruch 11 oder 12, wobei der Chrominanzbewegungsvektor einen Nicht-Viertelpixelwert in wenigstens einer Dimension im Chrominanzraum hat.

15. Verfahren nach Anspruch 11, wobei:
der Chrominanzbewegungsvektor einen Ganzpixelwert in einer gegebenen Dimension hat, wenn der Leuchtdichtebewegungsvektor einen Viertelpixel-Offset in der gegebenen Dimension hat;
der Chrominanzbewegungsvektor einen Viertelpixel-Offset in der gegebenen Dimension hat, wenn der Leuchtdichtebewegungsvektor einen Halbpixel-Offset in der gegebenen Dimension hat; und
der Chrominanzbewegungsvektor einen Halbpixel-Offset in der gegebenen Dimension hat, wenn der Leuchtdichtebewegungsvektor einen Dreiviertelpixel-Offset in der gegebenen Dimension hat.

16. Verfahren nach Anspruch 11, 12 oder 13, wobei der Leuchtdichtebewegungsvektor einen Halbpixelwert in einer Dimension im Leuchtdichteraum hat und wobei der Chrominanzbewegungsvektor einen entsprechenden Viertelpixelwert in der einen Dimension im Chrominanzraum hat.

17. Computerlesbares Medium, das computerausführbare Befehle speichert, um das Computersystem (**300**) dazu zu veranlassen, eines der Verfahren der Ansprüche 1 bis 16 während einer Videokodierung durchzuführen.

18. Computerlesbares Medium, das computerausführbare Befehle speichert, um das Computersystem (**300**) dazu zu veranlassen, eines der Verfahren der Ansprüche 1 bis 16 während einer Videodekodierung durchzuführen.

19. System zur Videokodierung oder -dekodierung (**300**) umfassend:
eine Einrichtung zum Auswählen (**1610**) eines von mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren;

eine Einrichtung zum Berechnen („computing") von Leuchtdichtebewegungsvektoren (Helligkeitsbewegungsvektoren; „luminance motion vectors") für einen vorhergesagten Videorahmen bezüglich eines Referenzvideorahmens;
eine Einrichtung zum Konvertieren der Leuchtdichtebewegungsvektoren in Chrominanzbewegungsvektoren gemäß des ausgewählten Rundungsmodus' für Chrominanzbewegungsvektoren,
wobei im Vergleich zu einem ersten Modus (**1620**) der mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren ein zweiter Modus (**1630**) der mehreren verfügbaren Rundungsmodi für Chrominanzbewegungsvektoren die Chrominanzbewegungsvektorpräzision verringert und dadurch die Berechnungskomplexität („computational complexity") nachfolgender Bewegungskompensation reduziert:

20. System (**300**) nach Anspruch 19, wobei das Auswählen des einen der mehreren verfügbaren Rundung smodi für Chrominanzbewegungsvektoren auf Basis einer Benutzereingabe durchgeführt wird.

21. System (**300**) nach Anspruch 19 oder 20, wobei wenigstens manche der Leuchtdichtebewegungsvektoren viertelpixelgenau sind und wobei der ausgewählte Rundungsmodus für Chrominanzbewegungsvektoren wenigstens manche der Leuchtdichtebewegungsvektoren auf viertelpixelgenaue Chrominanzbewegungsvektoren im Chrominanzraum abbildet.

22. System (**300**) nach Anspruch 19 oder 20, wobei wenigstens manche der Leuchtdichtebewegungsvektoren viertelpixelgenau sind und wobei der ausgewählte Rundungsmodus für Chrominanzbewegungsvektoren keinen der Leuchtdichtebewegungsvektoren auf viertelpixelgenaue Chrominanzbewegungsvektoren im Chrominanzraum abbildet.

23. System (**300**) nach einem der Ansprüche 19 bis 22, wobei ein Signal in einem Bitstrom den ausgewählten Rundungsmodus für Chrominanzbewegungsvektoren angibt.

24. System (**300**) nach Anspruch 23, wobei das Signal ein Ein-Bit-Wert in einem Sequenzlevelfeld des Bitstroms ist.

25. System nach Anspruch 19, wobei die Einrichtung zum Berechnen der Leuchtdichtebewegungsvektoren dazu eingerichtet ist, die Leuchtdichtebewegungsvektoren mit Viertelpixel-Genauigkeit für den vorhergesagten Videorahmen bezüglich des Referenzvideorahmens zu berechnen; und wobei die Einrichtung zum Konvertieren der Leuchtdichtebewegungsvektoren in die Chrominanzbewegungsvektoren dazu eingerichtet ist, die Leuchtdichtebewegungsvektoren in die Chrominanzbewegungsvektoren mit Viertelpixel-Genauigkeit im Chrominanzraum zu konvertieren.

26. System (**300**) nach Anspruch 25, wobei die Skala für die Chrominanzbewegungsvektoren eine halbe Skala der Leuchtdichtebewegungsvektoren ist.

27. System (**300**) nach Anspruch 25 oder 26, wobei die Einrichtung zum Konvertieren Chrominanzbewegungsvektoren, die einen Nicht-Viertelpixelwert haben, auf Halbpixel- oder Ganzpixelwerte rundet.

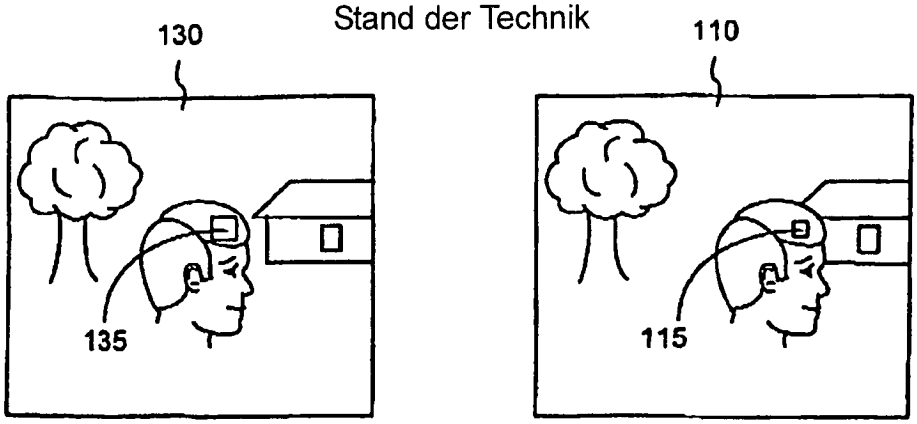
28. System (**300**) nach Anspruch 25 oder 26, wobei für einen gegebenen Leuchtdichtebewegungsvektor der Leuchtdichtebewegungsvektoren und entsprechenden Chrominanzbewegungsvektor der Chrominanzbewegungsvektoren:
der Chrominanzbewegungsvektor einen ganzzahligen Pixelwert in einer gegebenen Dimension hat, wenn der Leuchtdichtebewegungsvektor einen Viertelpixel-Offsetwert in der gegebenen Dimension hat;
der Chrominanzbewegungsvektor einen Viertelpixel-Offsetwert in der gegebenen Dimension hat, wenn der Leuchtdichtebewegungsvektor einen Halbpixel-Offsetwert in der gegebenen Dimension hat; und
der Chrominanzbewegungsvektor einen Halbpixel-Offsetwert in der gegebenen Dimension hat, wenn der Leuchtdichtebewegungsvektor einen Dreiviertelpixel-Offsetwert in der gegebenen Dimension hat.

Es folgen 13 Blatt Zeichnungen

Anhängende Zeichnungen

Fig. 1

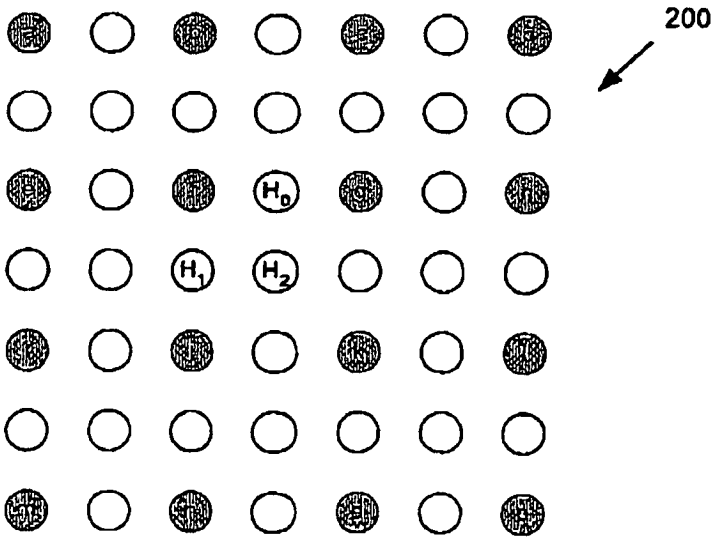
Stand der Technik



Referenzrahmen Prädiktionsrahmen

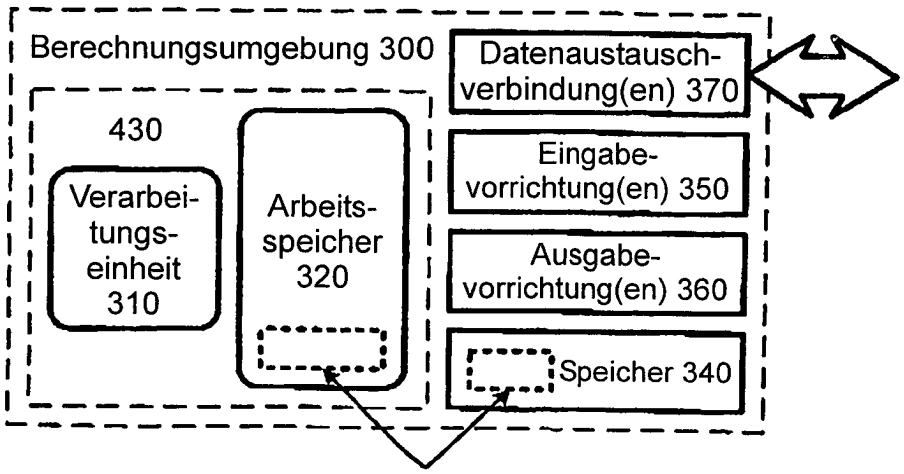
Fig. 2

Stand der Technik



- **a, b, c,..., p** bezeichnen Vollpixelstellen
- **H_0, H_1, H_2** bezeichnen Halbpixelstellen

Fig. 3



Software 380, die Subpixelinterpolationstechniken implementiert

Fig. 4

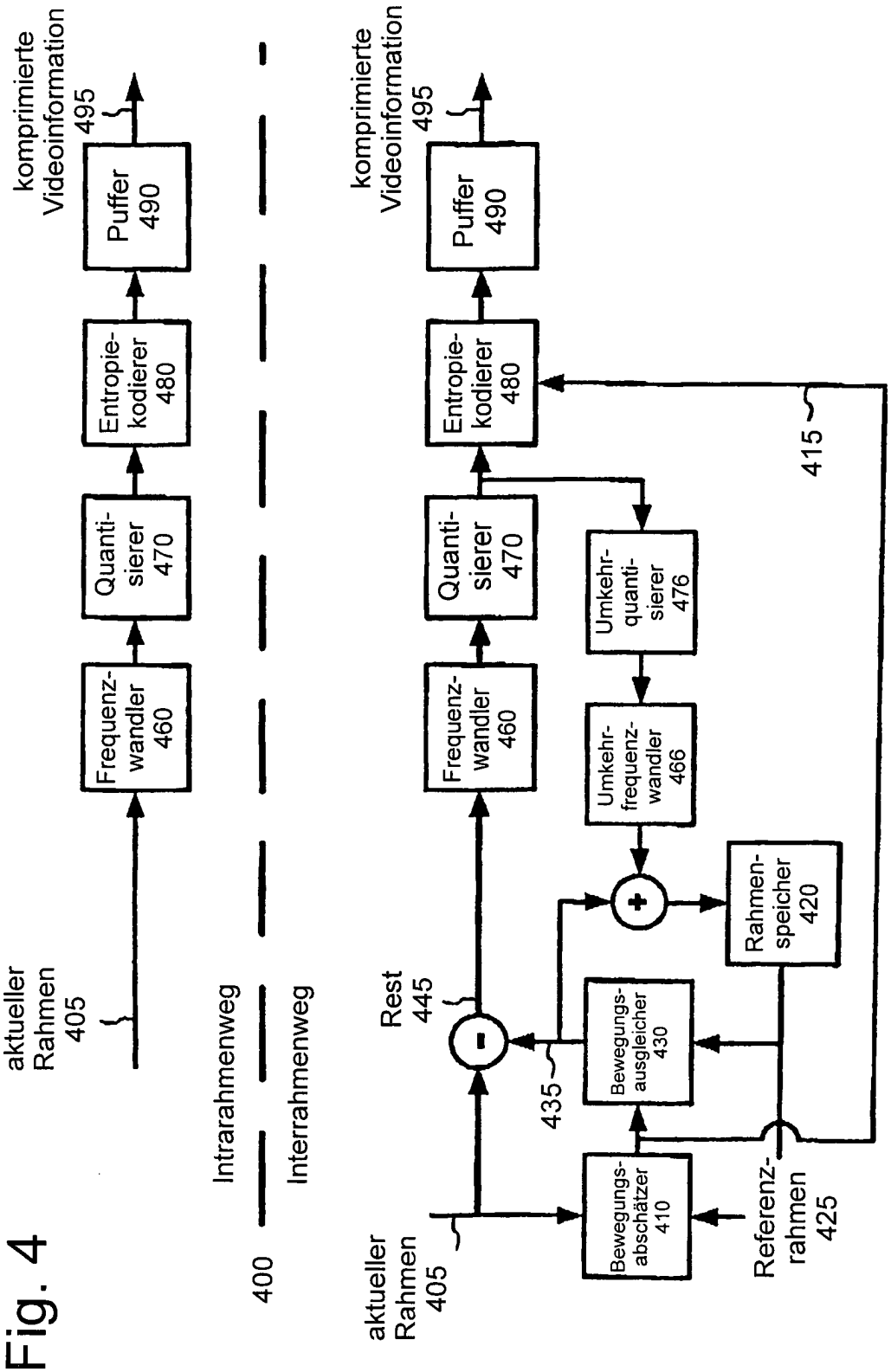

komprimierte Videoinformation 495
Puffer 490
Entropie-kodierer 480
Quanti-sierer 470
Frequenz-wandler 460
aktueller Rahmen 405
Intrarahmenweg
Interrahmenweg
400
Rest 445
Bewegungs-abschätzer 410
Bewegungs-ausgleicher 430
435
Umkehr-frequenz-wandler 466
Umkehr-quanti-sierer 476
Rahmen-speicher 420
Referenz-rahmen 425
415

Fig. 5

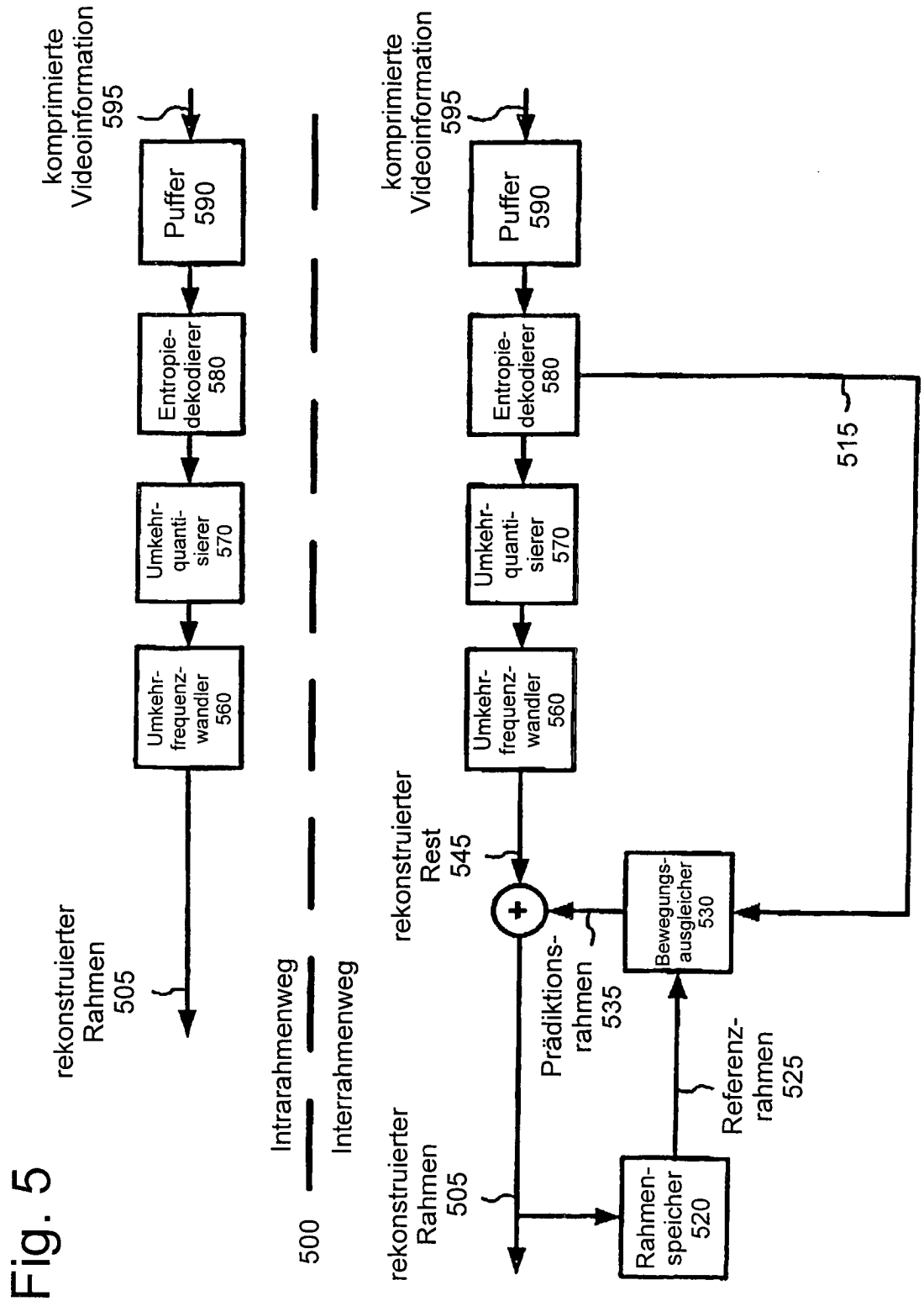
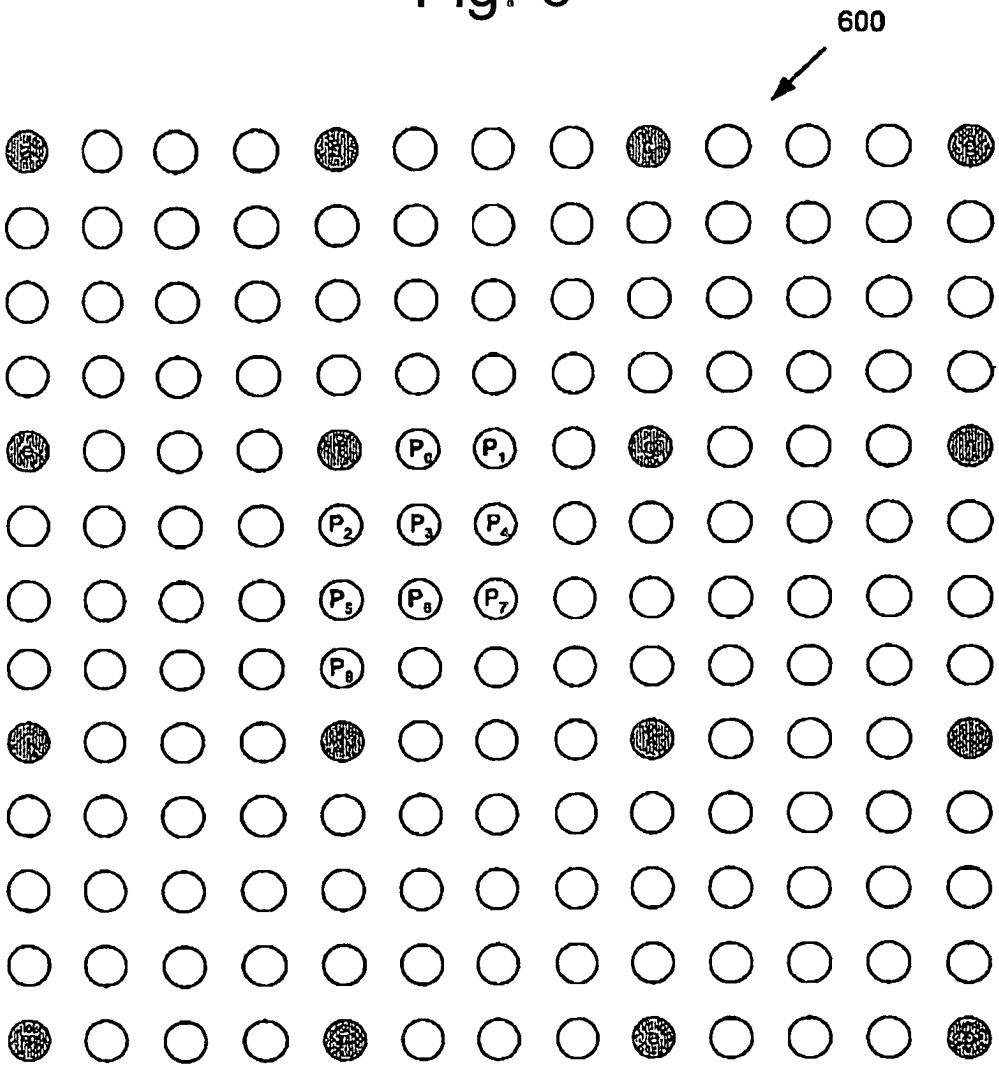

komprimierte Videoinformation 595
Puffer 590
Entropie-dekodierer 580
Umkehr-quanti-sierer 570
Umkehr-frequenz-wandler 560
rekonstruierter Rahmen 505
Intrarahmenweg
Interrahmenweg
500
komprimierte Videoinformation 595
Puffer 590
Entropie-dekodierer 580
515
Umkehr-quanti-sierer 570
Umkehr-frequenz-wandler 560
rekonstruierter Rest 545
Bewegungs-ausgleicher 530
Prädiktions-rahmen 535
Referenz-rahmen 525
rekonstruierter Rahmen 505
Rahmen-speicher 520

Fig. 6



- Jeder Kreis bezeichnet eine Pixelstelle in Viertelpixelintervallen
- **a . . . p** bezeichnen Vollpixelstellen
- **P_0 . . . P_8** bezeichnen Beispiele für Subpixelstellen

Fig. 7

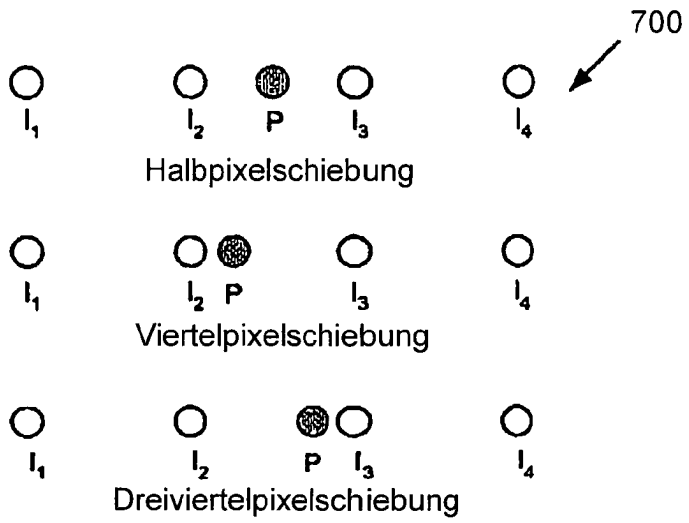

700
I1
I2
P
I3
I4
Halbpixelschiebung
Viertelpixelschiebung
Dreiviertelpixelschiebung

Fig. 8

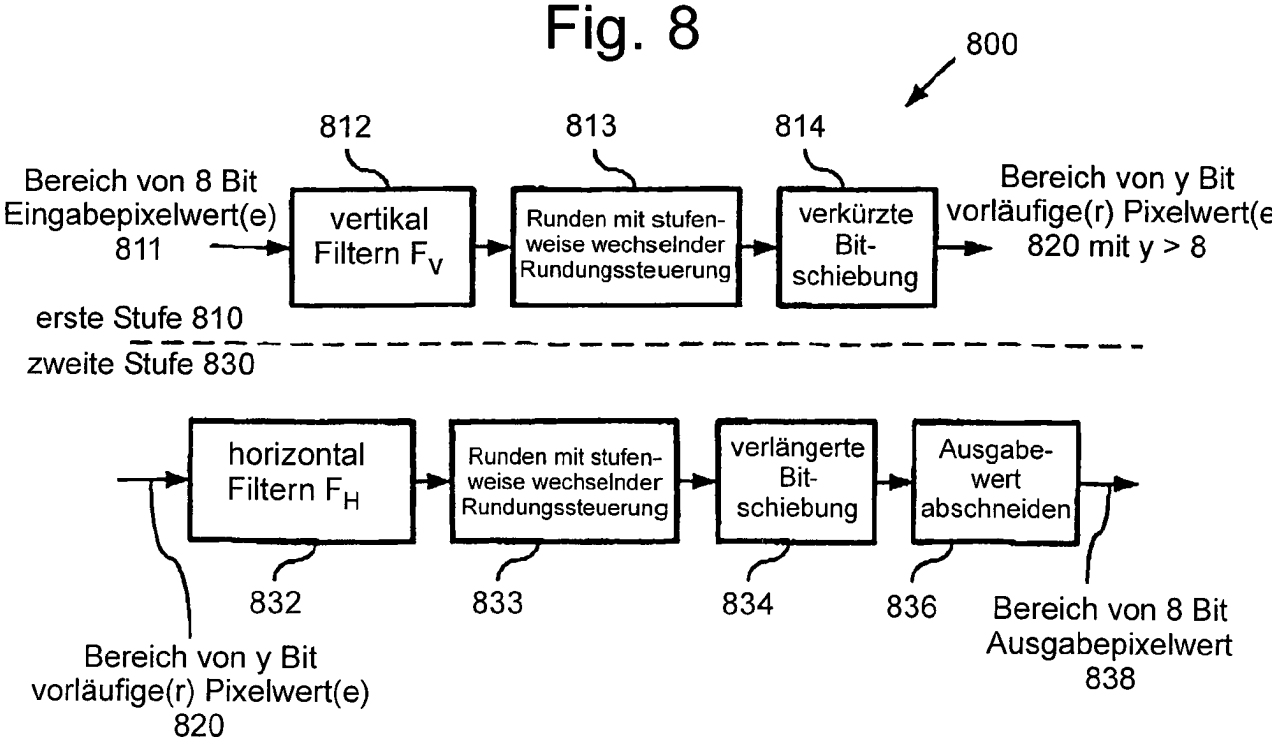

800
812
813
814
Bereich von 8 Bit Eingabepixelwert(e) 811
vertikal Filtern F_V
Runden mit stufenweise wechselnder Rundungssteuerung
verkürzte Bit-schiebung
Bereich von y Bit vorläufige(r) Pixelwert(e 820 mit y > 8
erste Stufe 810
zweite Stufe 830
horizontal Filtern F_H
Runden mit stufenweise wechselnder Rundungssteuerung
verlängerte Bit-schiebung
Ausgabewert abschneiden
832
833
834
836
Bereich von y Bit vorläufige(r) Pixelwert(e) 820
Bereich von 8 Bit Ausgabepixelwert 838

Fig. 9

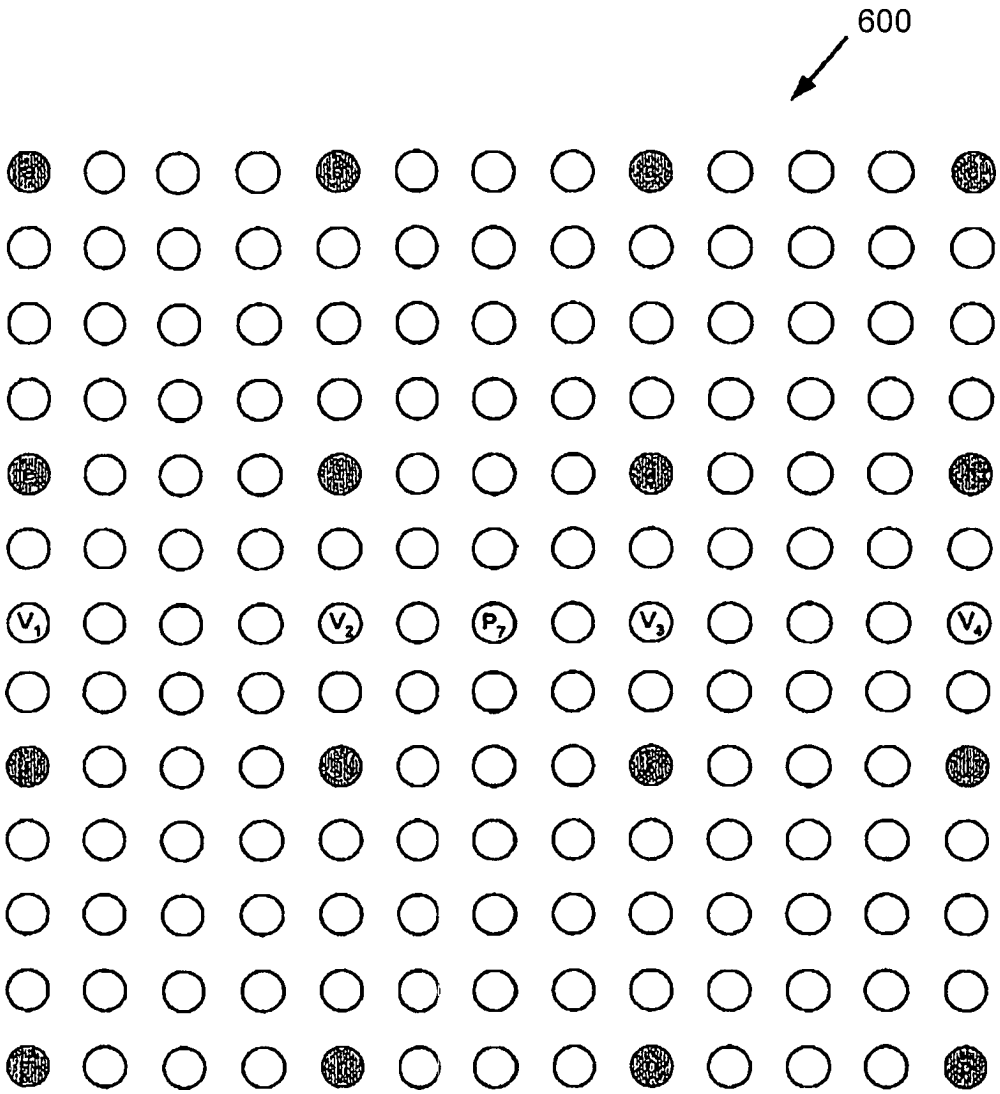


Fig. 10

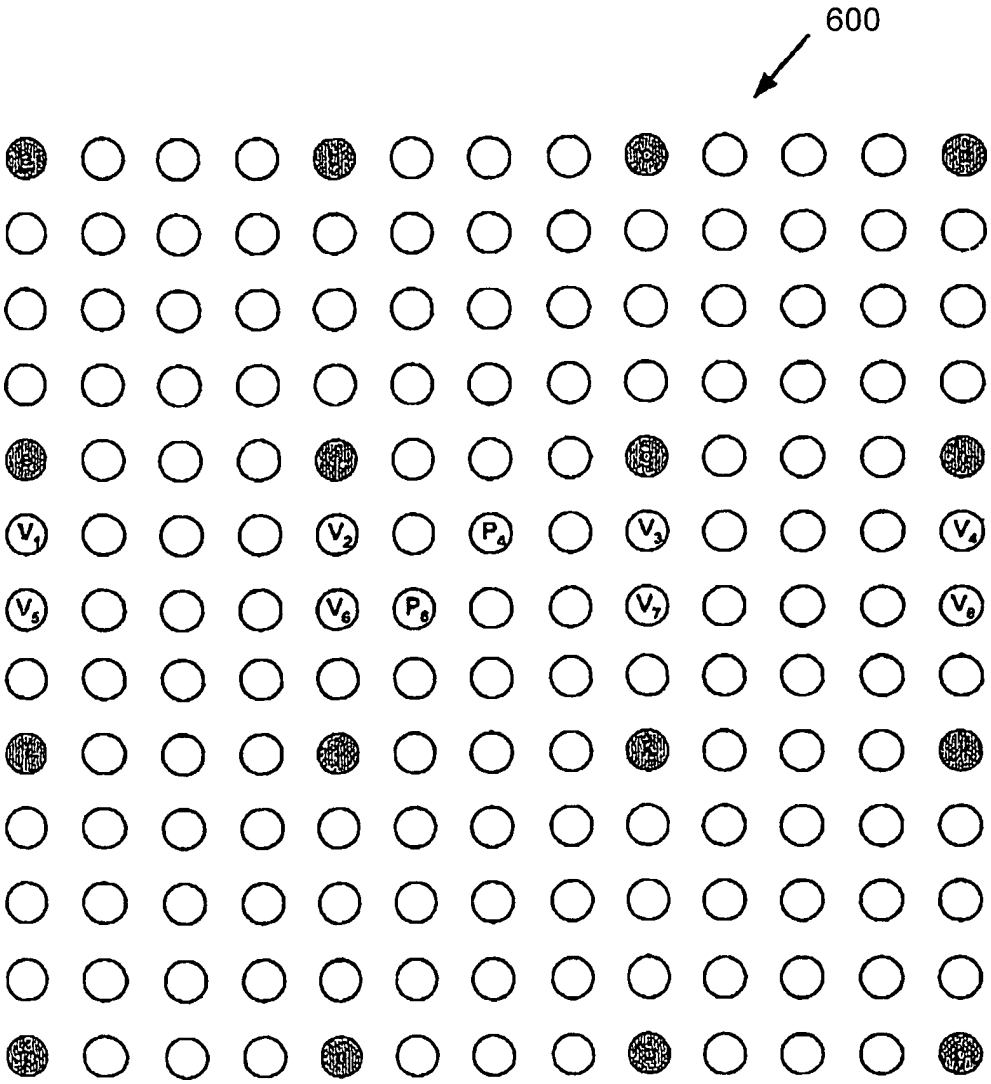


Fig. 11

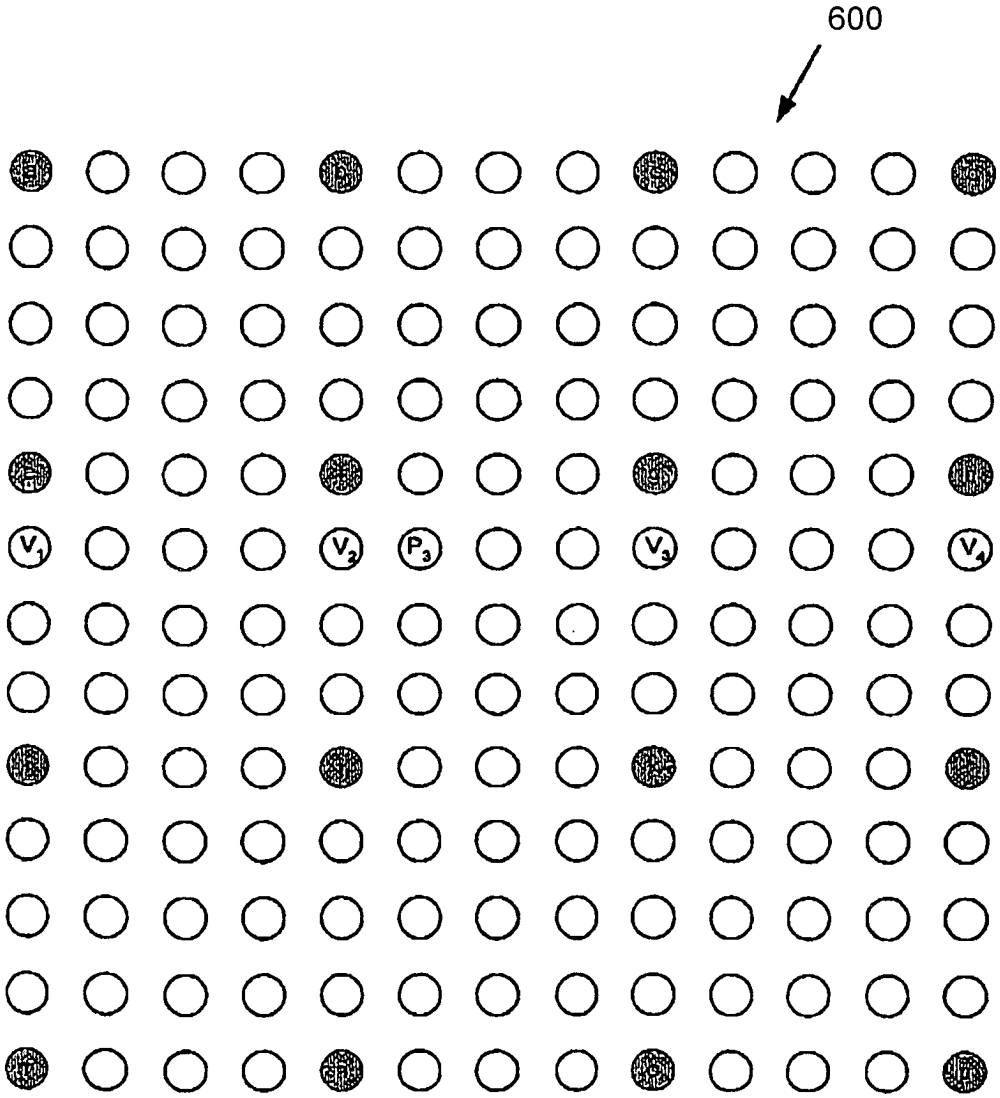


Fig. 12

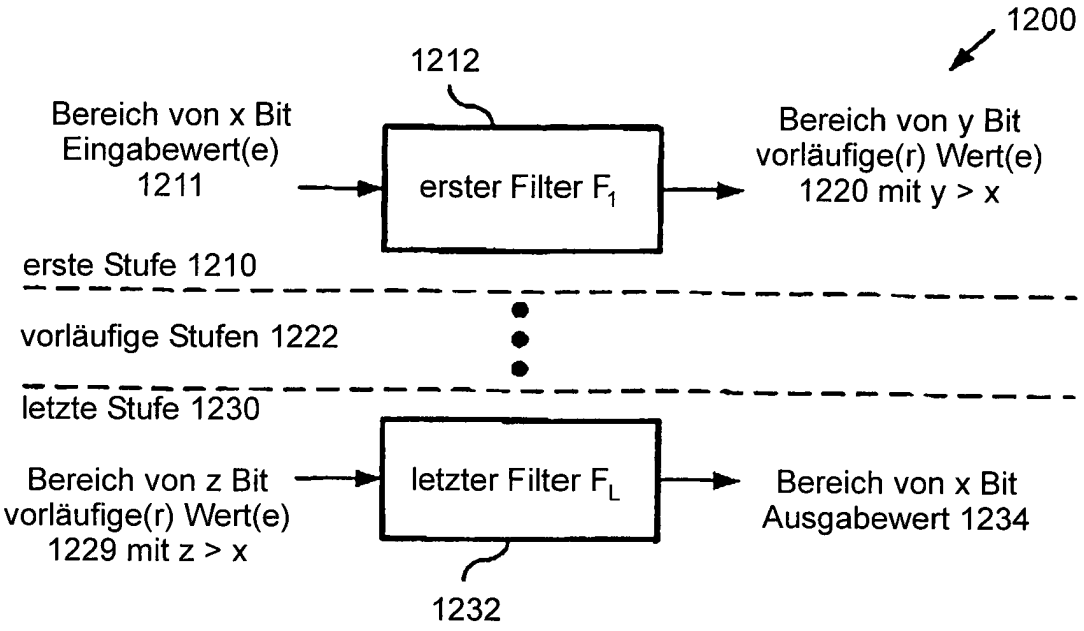

1200
1212
Bereich von x Bit
Eingabewert(e)
1211
erster Filter F_1
Bereich von y Bit
vorläufige(r) Wert(e)
1220 mit y > x
erste Stufe 1210
vorläufige Stufen 1222
letzte Stufe 1230
Bereich von z Bit
vorläufige(r) Wert(e)
1229 mit z > x
letzter Filter F_L
Bereich von x Bit
Ausgabewert 1234
1232

Fig. 13

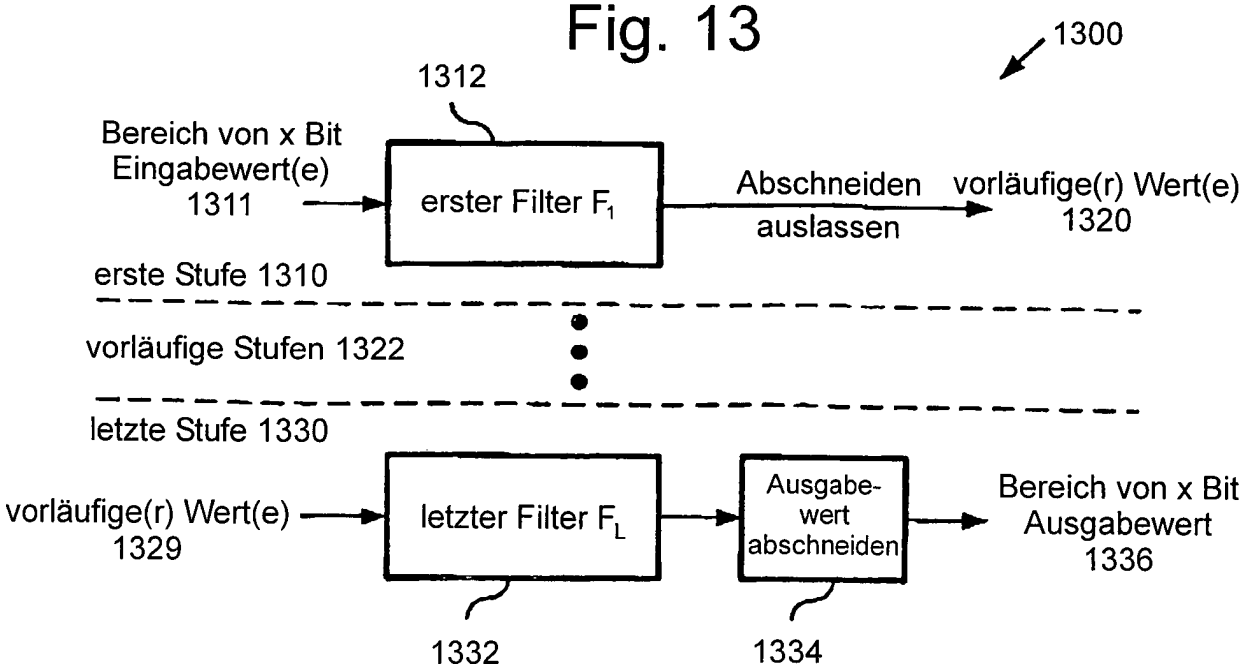

1300
1312
Bereich von x Bit
Eingabewert(e)
1311
erster Filter F_1
Abschneiden
auslassen
vorläufige(r) Wert(e)
1320
erste Stufe 1310
vorläufige Stufen 1322
letzte Stufe 1330
vorläufige(r) Wert(e)
1329
letzter Filter F_L
Ausgabe-
wert
abschneiden
Bereich von x Bit
Ausgabewert
1336
1332
1334

Fig. 14

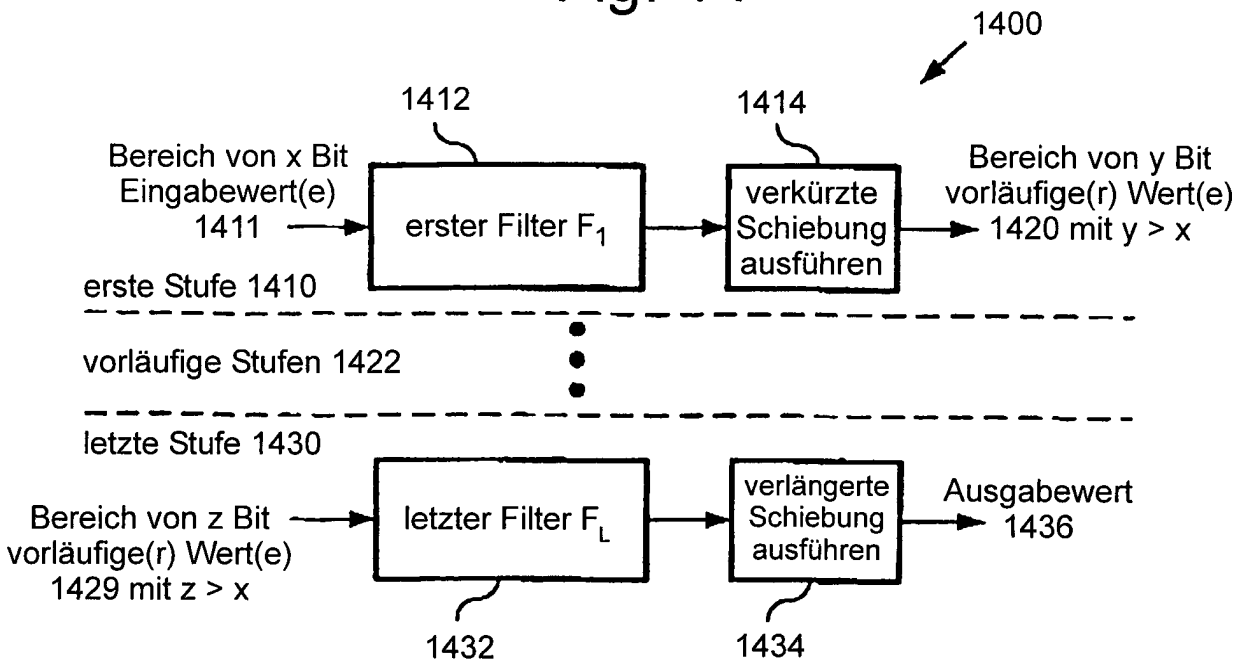

1400
1412
1414
Bereich von x Bit
Eingabewert(e)
1411
erster Filter F_1
verkürzte
Schiebung
ausführen
Bereich von y Bit
vorläufige(r) Wert(e)
1420 mit y > x
erste Stufe 1410
vorläufige Stufen 1422
letzte Stufe 1430
Bereich von z Bit
vorläufige(r) Wert(e)
1429 mit z > x
letzter Filter F_L
verlängerte
Schiebung
ausführen
Ausgabewert
1436
1432
1434

Fig. 15

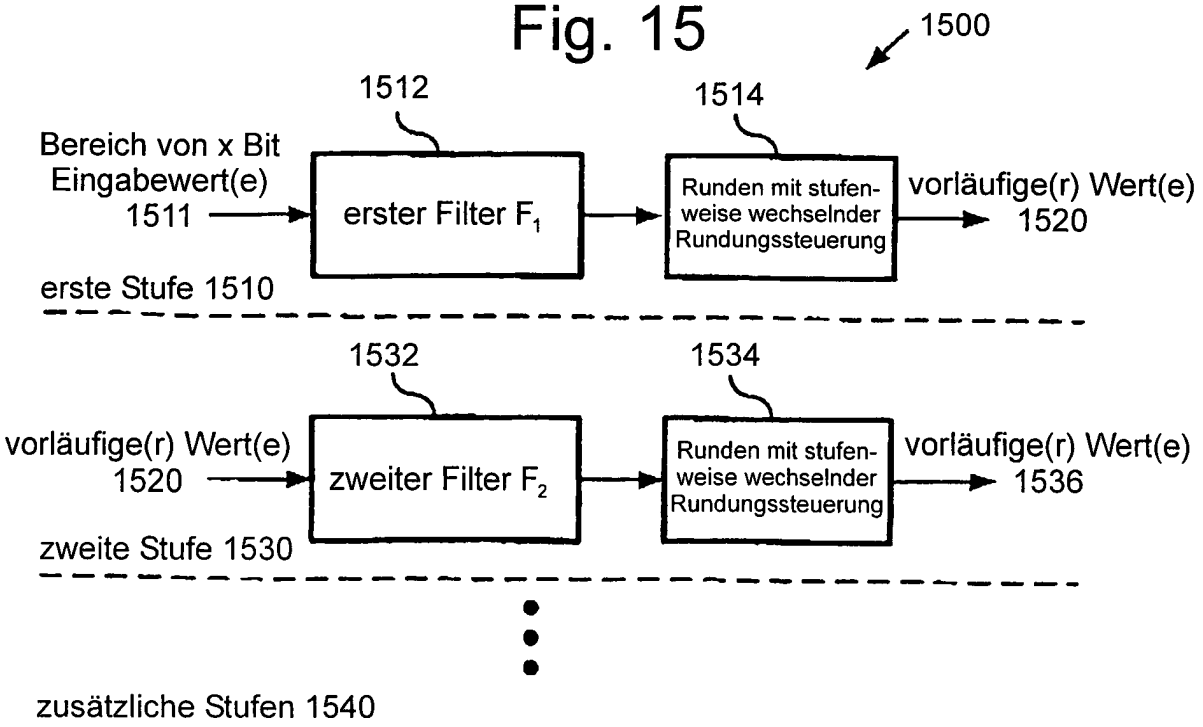

1500
1512
1514
Bereich von x Bit
Eingabewert(e)
1511
erster Filter F_1
Runden mit stufen-
weise wechselnder
Rundungssteuerung
vorläufige(r) Wert(e)
1520
erste Stufe 1510
1532
1534
vorläufige(r) Wert(e)
1520
zweiter Filter F_2
Runden mit stufen-
weise wechselnder
Rundungssteuerung
vorläufige(r) Wert(e)
1536
zweite Stufe 1530
zusätzliche Stufen 1540

Fig. 16

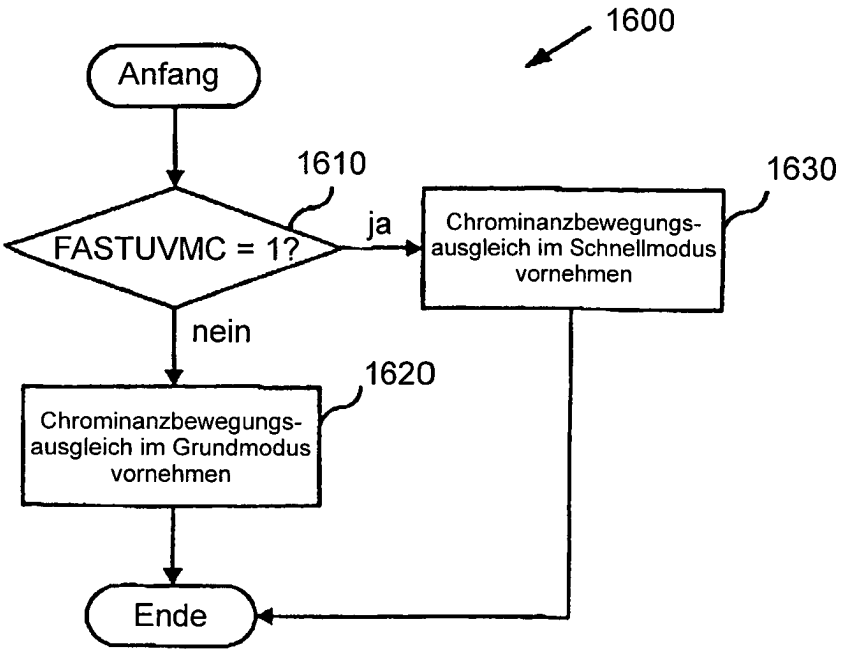
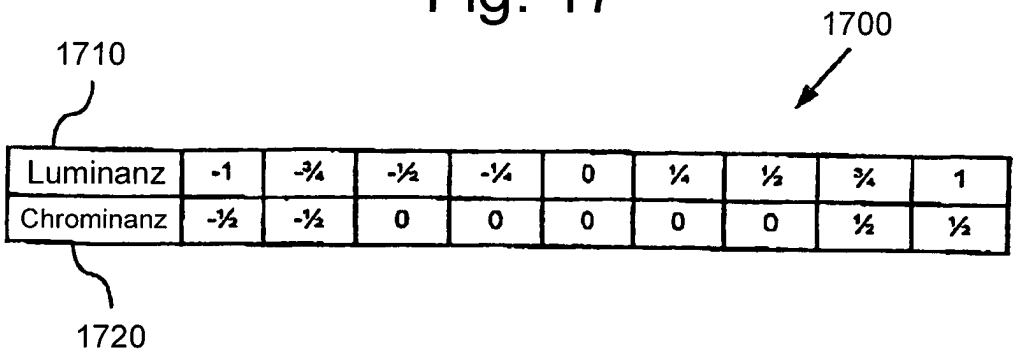
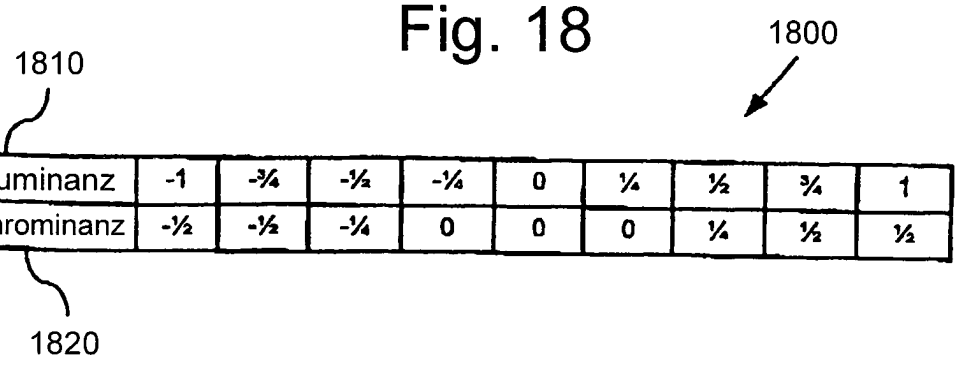

1600
Anfang
1610
FASTUVMC = 1?
ja
1630
Chrominanzbewegungs-ausgleich im Schnellmodus vornehmen
nein
1620
Chrominanzbewegungs-ausgleich im Grundmodus vornehmen
Ende

Fig. 17



Luminanz	-1	-¾	-½	-¼	0	¼	½	¾	1
Chrominanz	-½	-½	0	0	0	0	0	½	½

Fig. 18



Luminanz	-1	-¾	-½	-¼	0	¼	½	¾	1
Chrominanz	-½	-½	-¼	0	0	0	¼	½	½