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

(10) **DE 699 37 462 T2** 2008.02.07

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 135 934 B1**
(21) Deutsches Aktenzeichen: **699 37 462.6**
(86) PCT-Aktenzeichen: **PCT/US99/28276**
(96) Europäisches Aktenzeichen: **99 965 061.7**
(87) PCT-Veröffentlichungs-Nr.: **WO 2000/033583**
(86) PCT-Anmeldetag: **30.11.1999**
(87) Veröffentlichungstag der PCT-Anmeldung: **08.06.2000**
(97) Erstveröffentlichung durch das EPA: **26.09.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **31.10.2007**
(47) Veröffentlichungstag im Patentblatt: **07.02.2008**

(51) Int Cl.[8]: ***H04N 7/50*** (2006.01)
H03M 7/40 (2006.01)

(30) Unionspriorität:
201272 **30.11.1998** **US**

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

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

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

(72) Erfinder:
LIN, Chih-Lung (Bruce), Redmont, WA 98052, US;
LEE, Ming-Chieh, Bellevue, WA 98006, US

(54) Bezeichnung: **EFFIZIENTE MACROBLOCKHEADERKODIERUNG ZUR VIDEOKOMPRESSION**

Beschreibung

TECHNISCHES GEBIET

[0001] Die Erfindung bezieht sich auf Videocodierung (das Videocodieren), und im Speziellen auf ein verbessertes Verfahren zum Codieren von Blockparametern, die bei rahmenbasierten und objektbasierten Videocodierungsformaten verwendet werden.

HINTERGRUND

[0002] Anzeigegeräte für bewegte Bilder (full-motion video displays), die auf analogen Videosignalen basieren, sind schon lange in der Form von Fernsehgeräten verfügbar gewesen. Mit den kürzlich gemachten Fortschritten bei den Fähigkeiten der Computerverarbeitung und deren Erschwinglichkeit werden Anzeigegeräte für bewegte Bilder, basierend auf digitalen Videosignalen, weitreichender verfügbar. Digitale Videosysteme können signifikante Verbesserungen gegenüber konventionellen analogen Videosystemen beim Erzeugen, Modifizieren, Übertragen, Speichern und Abspielen von Bewegtbildsequenzen (full-motion video sequences) bieten.

[0003] Digitale Videoanzeigegeräte schließen eine große Anzahl von Bild-Frames (Bildrahmen) ein, die bei Frequenzen zwischen 30 und 75 Hz nacheinander abgespielt oder gerendert (wiedergegeben/sichtbar gemacht) werden. Jeder Bild-Frame ist ein Standbild, das von einem Bereich an Pixeln basierend auf der Auflösung des Anzeigegerätes eines bestimmten Systems gebildet wird. Als Beispiele haben VHS-basierte Systeme Anzeigeauflösungen von 320×480 Pixel, NTSC-basierte Systeme Anzeigeauflösungen von 720×486 Pixel und High-Definition-Fernsehsysteme (HDTV-Systeme), die in der Entwicklung stecken, Anzeigeauflösungen von 1360×1024 Pixel.

[0004] Die Mengen an rohen Digitalinformationen, die in Videosequenzen enthalten sind, sind massiv. Die Speicherung und Übertragung dieser Menge an Videoinformation ist mit konventioneller Personalcomputerausrüstung nicht durchführbar. Wenn man sich z.B. vorstellt, dass eine digitalisierte Form eines VHS-Bildformats mit einer relativ geringen Auflösung eine Auflösung von 320×480 Pixel aufweist. Ein Film in voller Länge mit einer Dauer von 2 Stunden entspricht bei dieser Auflösung 100 Gigabytes digitaler Videoinformation. Als Vergleich haben konventionelle optische Kompaktdisks eine Kapazität von ungefähr 0,6 Gigabytes, magnetische Festplatten eine Kapazität von 1 bis 2 Gigabytes und optische Kompaktdisks, die gerade entwickelt werden, Kapazitäten von bis zu 8 Gigabytes.

[0005] Um sich mit den Einschränkungen beim Speichern oder Übertragen solch massiver Mengen an digitalen Informationen zu befassen, sind verschiedene Videokomprimierungsstandards oder Prozesse entwickelt worden, einschließlich MPEG-1, MPEG-2 und H.26X. Diese Videokomprimierungstechniken nutzen Ähnlichkeiten zwischen nachfolgenden Bild-Frames aus, die als zeitliche oder Inter-Frame-Korrelation (Zusammenhänge) bezeichnet werden, um eine Inter-Frame-Komprimierung bereitzustellen, in der Bewegungsdaten und Fehlersignale verwendet werden, um die Veränderungen zwischen Frames zu codieren.

[0006] Zusätzlich nutzen die konventionellen Videokomprimierungstechniken Ähnlichkeiten zwischen Bild-Frames, die als räumliche oder Intra-Frame-Korrelation bezeichnet werden, um eine Intra-Frame-Komprimierung bereitzustellen, in der die Bild-Sample innerhalb eines Bild-Frames komprimiert werden. Intra-Frame-Komprimierung basiert auf konventionellen Prozessen zum Komprimieren von Standbildern, wie z.B. dem Codieren mit diskreter Kosinustransformation (DCT). Dieser Typ des Codierens wird manchmal auch als "Textur"- oder "Transformations"-Codierung bezeichnet. Eine "Textur" bezieht sich im Allgemeinen auf ein zweidimensionales Gebiet (array) mit Bild-Sample-Werten, wie z.B. ein Bereich mit Chrominanz- und Helligkeitswerten (luminance values) oder ein Gebiet mit α-Werten (Opazitäts-Werten). Der Begriff "Transformation" bezieht sich in diesem Kontext darauf, wie die Bild-Sample in räumliche Frequenzkomponenten während des Codierungsprozesses transformiert werden. Diese Verwendung des Begriffes "Transformation" sollte von einer geometrischen Transformation unterschieden werden, die zum Abschätzen von Szenenveränderungen in manchen Inter-Frame-Komprimierungsverfahren verwendet wird.

[0007] Inter-Frame-Komprimierung nutzt üblicherweise Bewegungsabschätzung und -kompensation, um Szenenveränderungen zwischen Frames zu codieren. Bewegungsabschätzung (motion estimation) ist ein Prozess zum Abschätzen der Bewegung eines Bild-Samples (z.B. Pixeln) zwischen Frames. Wenn Bewegungsabschätzung verwendet wird, versucht der Codierer Blöcke mit Pixeln in einem Frame mit entsprechenden Pixeln in einem anderen Frame in Übereinstimmung zu bringen. Nachdem der ähnlichste Block in einem gege-

benen Suchgebiet gefunden worden ist, wird die Positionsveränderung der Pixelorte der entsprechenden Pixel angenähert und als Bewegungsdaten, wie z.B. einem Bewegungsvektor, dargestellt. Bewegungskompensation ist ein Prozess zum Ermitteln eines vorhergesagten Bildes und zum Berechnen des Fehlers zwischen dem vorhergesagten Bild und dem echten Bild. Wenn Bewegungskompensation verwendet wird, wendet der Codierer die Bewegungsdaten auf ein Bild an und berechnet ein vorhergesagtes Bild. Der Unterschied zwischen dem vorhergesagten Bild und dem eingegebenen Bild wird als das Fehlersignal bezeichnet. Weil das Fehlersignal nur ein Array mit Werten ist, die den Unterschied zwischen Bild-Sample-Werten darstellen, kann es unter Verwendung desselben Textur-Codierungsverfahrens komprimiert werden, wie es für das Intra-Frame-Codieren von Bild-Samplen verwendet wurde.

[0008] Obwohl sie sich in bestimmten Implementationen unterscheiden, sind die Videokomprimierungsstandards MPEG-1, MPEG-2 und H.26X in vielerlei Hinsicht ähnlich. Die nachfolgende Beschreibung des MPEG-2-Videokomprimierungsstandards ist im Allgemeinen auf die anderen anwendbar.

[0009] MPEG-2 stellt Inter-Frame-Komprimierung und Intra-Frame-Komprimierung, basierend auf quadratischen Blöcken oder Arrays mit Pixeln in Videobildern, bereit. Ein Videobild wird in Bild-Sample-Blöcke geteilt, so genannte Makroblöcke, die eine Größenordnung von 16×16 Pixeln aufweisen. In MPEG-2 umfasst ein Makroblock vier Helligkeitsblöcke (luminance blocks) (wobei jeder Block 8×8 Sample der Helligkeit (Y) hat) und zwei Chrominanzblöcke (jeweils einen 8×8-Sample-Block für Cb und Cr).

[0010] In MPEG-2 wird Inter-Frame-Codierung auf Makroblöcken durchgeführt. Ein MPEG-2-Codierer führt Bewegungsabschätzung und -Kompensation durch, um Bewegungsvektoren und Blockfehlersignale zu berechnen. für jeden Block M_N in einem Bild-Frame N wird eine Suche über das Bild eines nächsten nachfolgenden Video-Frames N + 1 oder eines direkt vorangehenden Bild-Frames N – 1 (d.h. bidirektional) durchgeführt, um die entsprechenden ähnlichsten Blöcke M_{N+1} oder M_{N-1} zu identifizieren. Der Ort des ähnlichsten Blocks relativ zu dem Block M_N wird mit einem Bewegungsvektor (DX, DY) codiert. Der Bewegungsvektor wird anschließend verwendet, um einen Block mit vorhergesagten Sample-Werten zu berechnen. Diese vorhergesagten Sample-Werte werden mit dem Block M_N verglichen, um das Blockfehlersignal zu ermitteln. Das Fehlersignal wird unter Verwendung eines Textur-Codierungsverfahrens komprimiert, wie z.B. dem Codieren mit diskreter Kosinustransformation (DCT).

[0011] Objekt-basierte Videocodierungstechniken sind als eine Verbesserung bezüglich der konventionellen Frame-basierten Codierungsstandards vorgestellt worden. Beim Objekt-basierten Codieren werden willkürlich geformte Bildmerkmale von den Frames in der Videosequenz unter Verwendung eines Verfahrens, das als "Segmentierung" bezeichnet wird, getrennt. Die Videoobjekte oder "Segmente" werden unabhängig voneinander codiert. Objekt-basiertes Codieren kann die Komprimierungsrate verbessern, weil es die Inter-Frame-Korrelation zwischen Videoobjekten in aufeinanderfolgenden Frames erhöht. Es ist ebenso für eine Vielfalt von Anwendungen vorteilhaft, die einen Zugriff auf und das Verfolgen von Objekten in einer Videosequenz benötigen.

[0012] In den Objekt-basierten Videocodierungsverfahren, die für den MPEG-4-Standard vorgeschlagen wurden, werden die Form, Bewegung und Textur von Videoobjekten unabhängig codiert. Die Form (Gestalt) von einem Objekt wird durch eine binäre oder α-Maske (numerische Maske) dargestellt, die den Rand des beliebig geformten Objekts in einem Video-Frame definiert. Die Bewegung eines Objektes ist ähnlich zu den Bewegungsdaten aus MPEG-2, mit der Ausnahme, dass sie auf ein beliebig geformtes Bild des Objektes angewandt wird, das aus einem rechteckigen Frame segmentiert worden ist. Bewegungsabschätzung und -kompensation wird eher auf Blöcken einer "Videoobjektebene" durchgeführt, als auf dem gesamten Frame. Die Videoobjektebene ist der Name für das geformte Bild eines Objekts in einem einzelnen Frame.

[0013] Die Textur eines Videoobjekts ist die Bild-Sample-Information in einer Videoobjektebene, die in die Form des Objekts fällt. Texturcodierung der Bild-Sample und Fehlersignale eines Objekts wird unter Verwendung ähnlicher Texturcodierungsverfahren durchgeführt, wie bei dem Frame-basierten Codieren. Zum Beispiel kann ein segmentiertes Bild in ein umfassendes (umgrenzendes) Rechteck, das aus Makroblocks gebildet wurde, eingepasst werden. Das rechteckige Bild, das durch das umfassende Rechteck gebildet wird, kann einfach wie ein rechteckiger Frame komprimiert werden, mit der Ausnahme, dass transparente Makroblöcke nicht codiert werden müssen. Teilweise transparente Blöcke werden codiert, nachdem die Teile des Blocks, die aus der Formgrenze des Objekts fallen, mit Sample-Werten nach einer Technik, die "Auffüllen" (padding) genannt wird, aufgefüllt werden.

[0014] Frame-basierte Codierungstechniken, wie z.B. MPEG-2 und H.26X, und Objekt-basierte Codierungs-

techniken, die für MPEG-4 vorgeschlagen wurden, sind sich dadurch ähnlich, dass sie Intra-Frame- und Inter-Frame-Codierung auf Makroblöcken durchführen. Jeder Makroblock schließt eine Reihe von Overhead-Parametern ein, die Informationen über den Makroblock bereitstellen. Als ein Beispiel zeigt Fig. 1 Makroblockparameter, die in dem Header (Kopf, Kopfdaten) von einem Inter-Frame-Makroblock verwendet werden. Der COD-Parameter (**10**) ist ein einzelnes Bit, das anzeigt, ob der Inter-Frame-Makroblock codiert ist. Im Wesentlichen zeigt dieses Bit an, ob der codierte Makroblock Bewegungsdaten und Textur-codierte Fehlerdaten einschließt, oder nicht. In den Fällen, wo Bewegungs- und Fehlersignaldaten gleich null sind, reduziert das COD-Bit die Information, die benötigt wird, um den Makroblock zu codieren, weil nur ein einzelnes Bit gesendet wird, anstatt zusätzliche Bits, die anzeigen, dass der Bewegungsvektor und die Texturdaten nicht codiert sind.

[0015] Zusätzlich zu dem COD-Bit schließt die Codierungssyntax für Makroblöcke codierte Blockparameter (CBP) ein, die anzeigen, ob die codierten Transformationskoeffizienten für Chrominanz und Helligkeit für den Makroblock übermittelt wurden. Wenn die Transformationskoeffizienten für einen Block alle null sind, dann gibt es keinen Grund, Texturdaten für den Block zu senden. Die codierten Blockparameter für Chrominanz (CBPC) sind zwei Bits, die anzeigen, ob codierte Texturdaten für jeden der zwei Chrominanzblöcke übermittelt wurden, oder nicht.

[0016] Die CBPC-Bits werden zusammen mit einem anderen Flag (Bitschalter), der Informationen über den Typ der Quantisierung für den Makroblock bereitstellt, codiert. Diese Flags werden kombiniert, um einen Parameter zu bilden, der MCBPC (**12**) genannt wird, und wobei MCBPC Entropie-codiert ist, wobei ein Entropie-Codierungsverfahren, wie z.B. nach Huffman oder ein arithmetisches Codieren verwendet wird.

[0017] Der Parameter, der als AC_Pred_flag (**14**) bezeichnet wird, ist ein Flag, das anzeigt, ob eine AC-Vorhersage in dem Makroblock verwendet wurde.

[0018] Das codierte Blockmuster (Coded Block Pattern) für Helligkeit (CBPY) (**16**) umfasst vier Bits, die anzeigen, ob codierte Texturdaten für jeden der vier Helligkeitsblöcke übermittelt wurden, oder nicht. Wie der MCBPC-Parameter, sind die CBPY-Flags ebenso Entropie-codiert, wobei entweder Huffman- oder arithmetische Codierung verwendet wird.

[0019] Nach dem CBPY-Parameter schließt der Makroblock codierte Bewegungsvektordaten ein (als Element **18** in Fig. 1 gezeigt). Den Bewegungsvektordaten folgend stellen die "Blockdaten" die codierten Texturdaten für den Makroblock dar (als Blockdaten **20** in Fig. 1 gezeigt).

[0020] Eine Beeinträchtigung des Codierungsansatzes, der in Fig. 1 dargestellt ist, ist, dass er CBPC- und CBPY-Flags getrennt voneinander codiert, und deshalb die Korrelation zwischen diesen Parametern nicht ausnutzt, um den Makroblock-Overhead zu reduzieren. Zusätzlich zieht er keine Vorteile aus der örtlichen Abhängigkeit der codierten Blockparameter.

[0021] US-A-5 400 075 bezieht sich auf einen adaptiven Codierer/Decodierer für variable Längen (variable length encoder/decoder) für das Codieren mit variabler Länge eines komprimierten Videosignals.

KURZFASSUNG

[0022] Es ist die Aufgabe der Erfindung, ein verbessertes Verfahren und System zum Codieren in einer Videocodierungsanwendung bereitzustellen.

[0023] Diese Aufgabe wird durch die Erfindung, wie in den unabhängigen Ansprüchen beansprucht, gelöst.

[0024] Bevorzugte Ausführungsformen werden durch die abhängigen Ansprüche definiert.

[0025] Die Erfindung stellt ein verbessertes Verfahren zum Codieren der Makroblock-Header-Parameter in Videocodierungsanwendungen bereit. Ein Aspekt der Erfindung ist ein Codierungsverfahren, das die Korrelation zwischen den codierten Blockparametern durch gemeinsames Codieren aller codierter Blockparameter mit einem einzelnen Code variabler Länge (variable length code) ausnutzt. Ein anderer Aspekt der Erfindung ist ein Codierungsverfahren, das die Vorteile aus der räumlichen Abhängigkeit zwischen den codierten Blockparametern von benachbarten Blöcken zieht.

[0026] In einer Implementierung der Erfindung werden die codierten Blockparameter für Helligkeit und Chrominanz eines Makroblocks in einem einzelnen, kombinierten Parameter für den Makroblock gebildet. Der kom-

binierte Parameter wird einem Code variabler Länge aus einer Tabelle mit Codes variabler Länge zugewiesen. Die Codierungstabelle wird basierend auf einer Zielbitrate (z.B. Internetanwendungen mit geringer Bitrate) und einer Zielklasse eines Videoinhalts (z.B. Video mit einem sprechenden Kopf) trainiert. Durch das gemeinsame Codieren der Helligkeits- und Chrominanzwerte nutzt der Codierer die Korrelation zwischen diesen Parametern in dem Makroblock aus.

[0027] Um die Codierungseffizienz weiter zu verbessern, verwendet die Implementierung eine Vorhersage, um den Vorteil aus der räumlichen Abhängigkeit der codierten Blockparameter von benachbarten Blöcken zu ziehen. Bevor der Code variabler Länge dem Blockparameter zugewiesen wird, werden manche der codierten Blockparameter, ausgehend von benachbarten Blöcken, vorhergesagt. Für Intra-Frame-Makroblöcke, z.B., berechnet der Codierer einen räumlich vorhergesagten Wert für jeden codierten Blockparameter für die Helligkeit. Dieser räumlich vorhergesagte Parameter bildet einen Teil des kombinierten Parameters für den Makroblock.

[0028] Zusätzliche Merkmale und Vorteile der Erfindung werden besser von der folgenden detaillierten Beschreibung und den anhängigen Zeichnungen einer Implementierung der Erfindung ersichtlich.

KURZE BESCHREIBUNG DER FIGUREN

[0029] Fig. 1 ist ein Diagramm, das ein Beispiel eines Makroblock-Header, der in einem Standard-Videocodierungsprozess verwendet wurde, darstellt.

[0030] Fig. 2 ist ein Blockdiagramm eines Videocodierers.

[0031] Fig. 3 ist ein Blockdiagramm eines Videodecodierers.

[0032] Fig. 4 ist ein Diagramm, das ein Beispiel eines verbesserten Makroblock-Header darstellt, in dem die codierten Blockparameter für Chrominanz und Helligkeit gemeinsam mit einem einzelnen Code variabler Länge codiert wurden.

[0033] Fig. 5 ist ein Ablaufdiagramm, das darstellt, wie eine Implementierung der Erfindung einen einzelnen Code variabler Länge für die codierten Blockparameter von I- und P-Frame-Makroblöcken berechnet.

[0034] Fig. 6 ist ein Diagramm, das vier Makroblöcke und ihre entsprechenden Helligkeits-(Y)-Blöcke darstellt.

[0035] Fig. 7 ist ein Diagramm, das ein Beispiel für die vertikalen und horizontalen Gradienten von codierten Blockparameterwerten für ausgewählte Helligkeitsblöcke in Fig. 6 zeigt.

[0036] Fig. 8 ist ein Ablaufdiagramm, das ein Verfahren zum Berechnen eines Vorhersagemoduls (Prädiktor) für codierte Blockparameter darstellt.

[0037] Fig. 9 ist ein Diagramm eines Computersystems, das als eine Arbeitsumgebung für eine Softwareimplementierung der Erfindung dient.

DETAILLIERTE BESCHREIBUNG

Einführung

[0038] Der erste Teil unterhalb bietet eine Beschreibung eines Videocodierers und -decodierers. Anschließende Teile beschreiben, wie das Codieren von Makroblock-Header-Parametern durch das Ausnutzen der Korrelation zwischen CBPC- und CBPY-Parametern und durch das Ziehen des Vorteils aus der räumlichen Abhängigkeit von codierten Blockparametern von benachbarten Blöcken verbessert wird.

[0039] Sowohl für Frame-basiertes als auch Objekt-basiertes Videocodieren nützlich, verbessert die Erfindung das Codieren von Makroblockparametern, ob die Makroblöcke Komponenten von beliebigen Videoobjekten, die aus einer Sequenz von Frames segmentiert wurden, sind oder von rechteckig geformten Bild-Frames sind. Objekt-basiertes Codieren verwendet Codierungsmodule für ähnliche Bewegung und Textur, wie sie auch im Frame-basierten Codieren verwendet werden. Zusätzlich schließen Objekt-basierte Codierer ebenso Formcodierungsmodule ein. Die Blocksyntax, die für die Erfindung relevant ist, ist sowohl in Frame-basiertem als auch Objekt-basiertem Codieren ähnlich. Während der Codierer und Decoder in dem nächsten Abschnitt als

Objekt-basiert beschrieben werden, bieten sie eine ausreichende Basis zum Erläutern, wie die Erfindung sowohl in Frame-basierten als auch Objekt-basierten Codierungsschematas implementiert wird.

Beschreibung eines Beispiel-Codierers und -Decodierers

[0040] Fig. 2 ist ein Blockdiagramm, das eine Implementierung eines Objekt-basierten Videocodierers darstellt. Die Eingabe **30** in den Codierer schließt eine Reihe von Objekten, deren Form-(Gestalt-)Information und umgrenzende Rechtecke ein. Die Form-Information ist daher verfügbar, bevor der Codierer Textur- oder Bewegungsdaten codiert. Frame-basiertes Codieren unterscheidet sich dadurch, dass der gesamte Frame ohne Form-Information codiert wird.

[0041] Das Formcodierungsmodul **32** empfängt die Definition eines Objektes, einschließlich seines umgrenzenden Rechtecks und erweitert das umgrenzende Rechteck zu einem ganzzahligen Vielfachen von Makroblöcken. Die Form-Information für ein Objekt umfasst eine Maske oder "α-Ebene". Das Formcodierungsmodul **32** liest diese Maske und komprimiert sie unter Verwendung eines, z.B., konventionellen Verkettungscodierungsverfahrens (chain coding method), um die Kontur des Objektes zu codieren.

[0042] Bewegungsabschätzungsmodul **34** liest ein Objekt einschließlich seines umgrenzenden Rechteckes und eines vorher rekonstruierten Bildes **36** und berechnet Bewegungsschätzdaten, die verwendet werden, um die Bewegung des Objektes von einem Frame zum anderen vorherzusagen. Nach dem Identifizieren der Makroblöcke in dem aktuellen Objektbild sucht das Bewegungsschätzmodul **34** nach dem ähnlichsten Makroblock in dem rekonstruierten Bild für jeden Makroblock in dem aktuellen Objektbild, um die Bewegungsdaten für jeden Makroblock zu berechnen. Das bestimmte Format der Bewegungsdaten von dem Bewegungsschätzmodul **34** kann abhängig von dem Bewegungsabschätzungsverfahren, das verwendet wird, variieren. Die Implementierung, die unterhalb beschrieben ist, berechnet einen Bewegungsvektor für jeden Makroblock, welcher mit derzeitigen MPEG- und H26X-Formaten konsistent ist.

[0043] Das Bewegungskompensationsmodul **38** liest die Bewegungsvektoren, die durch das Bewegungsabschätzungsmodul berechnet wurde, und das vorher rekonstruierte Bild **36** ein, und berechnet ein vorhergesagtes Bild für den aktuellen Frame. Der Codierer findet den Unterschied zwischen den Bild-Sample-Werten in dem eingegebenen Bildblock, wie er in der Eingabe **30** spezifiziert ist, und den entsprechenden Sample-Werten in dem vorhergesagten Bildblock, wie er durch das Bewegungskompensationsmodul **38** berechnet wurde, um das Fehlersignal für den Makroblock zu ermitteln.

[0044] Das Texturcodierungsmodul **40** komprimiert dieses Fehlersignal für Inter-Frame codierte Objekte und komprimiert Bild-Sample-Werte für das Objekt von dem eingegebenen Datenstrom **30** für Intra-Frame codierte Objekte. Der Feedback-Pfad **42** von dem Texturcodierungsmodul **40** stellt das decodierte Fehlersignal dar. Der Codierer verwendet die Fehlersignal-Makroblöcke zusammen mit den vorhergesagten Bild-Makroblöcken von dem Bewegungskompensationsmodul, um das vorher rekonstruierte Bild **36** zu berechnen.

[0045] Das Texturcodierungsmodul **40** codiert Blöcke von Intra-Frame- und Fehlersignaldaten für ein Objekt unter Verwendung irgendeiner von einer Vielfalt von Standbild-Komprimierungstechniken. Beispiel-Komprimierungstechniken schließen Transformations-basierte Techniken, wie z.B. DCT, und Wavelet-Codierung, sowie andere konventionelle Bildkomprimierungsverfahren ein, wie z.B. LaPlacian-Pyramiden-Codierung.

[0046] Der Bitstrom mit komprimierten Videosequenzen schließt die Form-, Bewegungs- und Textur-codierten Informationen von den Modulen für Form-Codierung, Bewegungsabschätzung und Texturcodierung ein. Der Multiplexer **44** kombiniert und formatiert diese Daten in einer geeigneten Syntax und gibt sie in den Puffer **46** aus.

[0047] Während der Codierer in Hardware oder Software implementiert werden kann, wird er höchstwahrscheinlich in Software implementiert. In einer Softwareimplementierung stellen die Module in dem Encoder Softwareinstruktionen, die in dem Speicher eines Computers gespeichert sind, und im Prozessor ausgeführt werden, und die Videodaten, die im Speicher gespeichert sind, dar. Ein Software-Encoder kann auf einer Vielfalt von konventionellen computerlesbaren Datenträgern gespeichert und verteilt werden. In Hardwareimplementierungen sind die Encodermodule in einer digitalen Logik implementiert, vorzugsweise in einem integrierten Schaltkreis. Manche der Codiererfunktionen können in einem Spezialzweckgerät mit digitaler Logik in einer Computerperipherie optimiert werden, um einem Host-Computer die Verarbeitungslast abzunehmen.

[0048] Fig. 3 ist ein Blockdiagramm, das einen Decoder für ein Objekt-basiertes Videocodierungsverfahren

darstellt. Ein Demultiplexer **60** empfängt einen Bitstrom **62**, der eine komprimierte Videosequenz und separate Form-, Bewegungs- und Textur-codierte Daten auf einer Objekt-nach-Objekt-Basis darstellt. Um dies zu erreichen, setzt er einen Formdecoder ein, der die Umkehrung des Formcodierungsverfahrens, das von dem Encoder aus Fig. 2 verwendet wurde, implementiert. Die resultierenden Formdaten sind eine Maske, wie z.B. eine binäre α-Ebene oder Graustufen-α-Ebene, die die Form des Objektes darstellen.

[0049] Das Bewegungsdecodierungsmodul **66** codiert die Bewegungsinformation in dem Bit-Datenstrom. Die decodierte Bewegungsinformation schließt Bewegungsdaten ein, wie z.B. Bewegungsvektoren für Blöcke in Makroblöcken oder geometrische Transformationskoeffizienten, abhängig von dem Typ des in dem Codierer verwendeten Abschätzungsverfahrens. Das Bewegungsdecodierungsmodul **66** stellt diese Bewegungsinformation an das Bewegungskompensationsmodul **68** bereit, und das Bewegungskompensationsmodul **68** wendet die Bewegungsdaten auf vorher rekonstruierte Objektdaten **70** an.

[0050] Das Texturdecodierungsmodul **74** decodiert Fehlersignale für Inter-Frame codierte Texturdaten und ein Array mit Farbwerten für Intra-Frame-Texturdaten und gibt diese Information an ein Modul **72** zur Berechnung und Akkumulation des rekonstruierten Bildes weiter. Für Inter-Frame-codierte Objekte wendet dieses Modul **72** die Fehlersignaldaten auf die vorhergesagte Bildausgabe von dem Bewegungskompensationsmodul an, um das rekonstruierte Objekt für den aktuellen Frame zu berechnen. Für Intra-Frame-codierte Objekte decodiert das Texturdecodierungsmodul **74** die Bild-Sample-Werte für das Objekt und platziert das rekonstruierte Objekt in dem Modul **72** für rekonstruierte Objekte. Bereits vorher rekonstruierte Objekte werden temporär in dem Objektspeicher **70** gespeichert und werden verwendet, um das Objekt für andere Frames zu konstruieren.

[0051] Wie der Codierer, so kann der Decodierer in Hardware, Software oder einer Kombination von beidem implementiert sein. In Softwareimplementationen sind die Module in dem Decoder Softwareinstruktionen, die in einem Speicher eines Computers gespeichert sind und durch den Prozessor ausgeführt werden, und Videodaten, die in dem Speicher gespeichert sind. Ein Softwaredecodierer kann auf einer Vielfalt von konventionellen computerlesbaren Datenträgern gespeichert und verteilt werden. In Hardwareimplementierungen sind die Decodierermodule in einer digitalen Logik, vorzugsweise einem integrierten Schaltkreis, implementiert. Manche der Decodierungsfunktionen können in Geräten mit digitaler Logik für einen speziellen Zweck in einem Computerperipheriegerät optimiert werden, um die Arbeitslast von einem Host-Computer zu nehmen.

Verbessertes Codieren des Makroblock-Overhead

[0052] Die Erfindung schließt Innovationen ein, die das Codieren von Makroblock-Header-Parametern verbessern. Eine Innovation ist ein Verfahren zum Codieren der codierten Blockparameter, um die Korrelation zwischen CBPC und CBPY auszunutzen. Diese Innovation wird durch das gemeinsame Codieren eines kombinierten CBPC- und CBPY-Parameters mit einem einzelnen Code variabler Länge implementiert. Eine andere Innovation verbessert des Weiteren die Codierungseffizienz der Header-Parameter durch das Ausnutzen der räumlichen Abhängigkeiten der codierten Blockparameter. Im Wesentlichen werden codierte Blockparameter durch das Vorhersagen dieser von dem Parameter der Nachbarblöcke effizienter komprimiert.

[0053] Fig. 4 ist ein Diagramm, das die Header-Blockparameter darstellt, die durch eine Implementierung dieser Erfindung berechnet wurden. Wie die Header-Information, die in Fig. 1 gezeigt ist, schließt dieser Header-Block einen COD-Parameter **80**, ein AC_Pred_flag **82**, Bewegungsvektordaten (motion vector data – MV **84**) und Blockdaten **86** ein. Im Gegensatz zu dem Header in Fig. 1 sind die MCBPC- und CBPY-Parameter gemeinsam mit einem einzelnen Code variabler Länge codiert, und wird MBCBPCY **88** genannt. Dieser Code kombiniert codierte Blockparameter für Chrominanz und Helligkeit, sowie das Flag für den Makroblocktyp.

[0054] Fig. 5 ist ein Ablaufdiagramm, das darstellt, wie die Implementation einen Code variabler Länge für Intra-(I)-Frames und vorhergesagte (predicted) (P)-Frames erzeugt. In dieser bestimmten Implementation werden die Header-Blöcke für I- und P-Frames unterschiedlich codiert. Für I-Frames führt der Encoder den zusätzlichen Schritt des Vorhersagens der codierten Blockparameter für Helligkeit durch, bevor ein Code variabler Länge ausgewählt wird. Es ist ebenso möglich, eine Vorhersage für P-Frames zu verwenden. Jedoch verbessert eine Vorhersage die Codierungseffizienz in P-Frames nicht signifikant, und in manchen Fällen kann die Codierungseffizienz sogar verschlechtert werden.

[0055] Das Ziel der Verwendung einer Vorhersage für codierte Blockparameter ist es, so viele Null-Werte für diese Parameter zu erzeugen, wie möglich. Durch das Erzeugen möglichst vieler Null-Werte reduziert der Encoder die Varianz der codierten Blockparameter. Der Prozess des Trainierens der Tabelle für Codes mit variabler Länge kann anschließend den Null-Wert favorisieren, was die Codierungseffizienz verbessert. In P-Fra-

mes, besonders bei Anwendungen mit geringer Bitrate, sind die codierten Blockparameter meist vor der Vorhersage null. Als solches tendiert die Vorhersage nicht dazu, die Anzahl der Null-Werte zu erhöhen, und manchmal senkt sie sogar die Anzahl der Null-Werte. Deshalb verwendet die Implementation, die in Fig. 5 gezeigt ist, keine Vorhersage für P-Frames.

[0056] Für P-Frames beginnt der Codierer mit dem Finden der codierten Blockparameter für Helligkeit und Chrominanz, wie es in Schritt **100** gezeigt ist. Diese Blockparameter sind jeweils ein einzelnes Bit, das anzeigt, ob ein entsprechender Block Textur-codiert ist. Die codierten Blockparameter werden in dem Texturcodierungsmodul (**40** in Fig. 2) berechnet, welches ein Flag für einen codierten Block für jeden Block setzt, der eine Anzahl codierter Texturwerte ungleich null hat. Im umgekehrten Fall ist der Wert des codierten Blockparameters für einen Block, in dem die Texturwerte alle null sind (oder so nahe an null, dass es vernachlässigbar ist), null.

[0057] Weil es zwei Blöcke für die Chrominanz (je einen für die U- und V-Blöcke mit 8×8 Pixeln) und vier Blöcke für die Helligkeit (je einen für die vier 8×8 Blöcke) in dem Makroblock gibt, hat der kombinierte Parameter für das codierte Blockmuster insgesamt 6 Bits. Durch das Kombinieren dieser 6-Bit-Nummer mit dem einzelnen Bit für den Makroblocktyp, bildet der Codierer eine 7-Bit-Nummer, wie es in Schritt **102** gezeigt ist. Der Makroblocktyp zeigt an, ob der Makroblock für einen I- oder P-Frame ist.

[0058] Sobald der kombinierte MBCBPCY gebildet wurde, wird der kombinierte Parameter in einer Codierungstabelle mit Codes variabler Länge nachgesehen, um einen entsprechenden Code variabler Länge, zugehörig zu dem Parameter zu finden, wie es in Schritt **104** gezeigt ist. Der Codierer weist dem kombinierten Parameter MBCBPCY einen einzelnen Code variabler Länge zu.

[0059] Die Codierungstabelle in der Implementationstabelle ist eine Huffman-Codierungstabelle. Die Tabelle wird vorzugsweise basierend auf der Zielrate und des Zielszenarios trainiert. Tabelle 1 unterhalb ist eine Codierungstabelle mit Codes variabler Länge (Variable Length Coding (VLC) table), die für ein Szenario mit einem "sprechenden Kopf" und geringer Bitrate erlangt wurde. Für jeden Makroblock in einem P-Frame wird die kombinierte MBCBPCY-Information unter Verwendung des Code-Wortes für den entsprechenden Eintrag in dieser Tabelle codiert.

Tabelle 1: VLC-Tabelle für codierte Blockmuster der Chrominanz und der Helligkeit für P-Bilder

Index	MB type	CBPCY Y (1234) UV	Number of bits	Code
0	I	000000	7	1000000
1	I	000001	13	1001111001001
2	I	000010	12	100111111101
3	I	000011	15	000000111111100
4	I	000100	12	100111111100
5	I	000101	18	000000101010000011
6	I	000110	17	10010110100110100
7	I	000111	16	1000001110111100

Index	MB type	CBPCY Y (1234) UV	Number of bits	Code
8	I	001000	12	100000111010
9	I	001001	17	00000011111111000
10	I	001010	16	0000001111111101
11	I	001011	16	0000001111111111
12	I	001100	13	0000001111001

13	I	001101	18	000000101010000010
14	I	001110	16	1001011010011101
15	I	001111	16	0000001010100100
16	I	010000	12	100101111000
17	I	010001	17	00000010101000011
18	I	010010	15	100000111011111
19	I	010011	17	00000011111111001
20	I	010100	13	1001011110011
21	I	010101	18	100101101001101011
22	I	010110	18	100101111011111001
23	I	010111	16	0000001111111010
24	I	011000	14	10000011101110
25	I	011001	20	10010110100110101011
26	I	011010	16	1001011010011100
27	I	011011	18	100101111011111000
28	I	011100	13	1001011010010
29	I	011101	18	000000101010000101
30	I	011110	16	1001011010011110
31	I	011111	15	100101111001000
32	I	100000	12	000000111101
33	I	100001	17	10010111101111111
34	I	100010	16	0000001010100010
35	I	100011	16	1001011010011111
36	I	100100	14	10010111101110
37	I	100101	21	100101101001101010101
38	I	100110	17	100101111011111101
39	I	100111	17	100101111011111110
40	I	101000	12	100111100101
41	I	101001	18	000000101010000001

42	I	101010	19	100101101001101 0100
43	I	101011	16	100000111011110 1
44	I	101100	13	0000001111000
45	I	101101	16	100101101001101 1
46	I	101110	16	0000001111111110
47	I	101111	16	000000101010010 1
48	I	110000	13	0000001111110
49	I	110001	18	000000101010000 000
50	I	110010	16	000000101010001 1
51	I	110011	16	0000001111111011
52	I	110100	13	1000001110110
53	I	110101	18	000000101010000 100

Index	MB type	CBPCY Y (1234) UV	Number of bits	Code
54	I	110110	15	000000101010011
55	I	110111	15	100101111001001
56	I	111000	13	0000001010101
57	I	111001	21	100101101001101 010100
58	I	111010	15	100101111011110
59	I	111011	14	10010111100101
60	I	111100	10	1001011011
61	I	111101	15	100101101001100
62	I	111110	12	100101101011
63	I	111111	12	100101101010
64	P	000000	2	01
65	P	000001	7	0000000
66	P	000010	6	100110
67	P	000011	9	100101011
68	P	000100	3	111
69	P	000101	10	1000001111
70	P	000110	9	000000100
71	P	000111	12	000000101000
72	P	001000	3	110
73	P	001001	10	1000001010
74	P	001010	9	100101000
75	P	001011	12	000000101011
76	P	001100	5	10001
77	P	001101	11	00000011011
78	P	001110	9	100111010

79	P	001111	11	10011111111
80	P	010000	4	0011
81	P	010001	10	1001110111
82	P	010010	9	100000110
83	P	010011	12	100000111001
84	P	010100	4	1011
85	P	010101	10	1001111011
86	P	010110	9	100101100
87	P	010111	11	10010111111
88	P	011000	6	001001
89	P	011001	12	000000110101
90	P	011010	10	1001111110
91	P	011011	13	1001111001000
92	P	011100	6	000001
93	P	011101	11	10010101010
94	P	011110	10	1000001000
95	P	011111	12	000000101001
96	P	100000	4	0001
97	P	100001	10	1001010100
98	P	100010	9	100101110
99	P	100011	12	100000111000

Index	MB type	CBPCY Y (1234) UV	Number of bits	Code
100	P	100100	6	100100
101	P	100101	11	10011110011
102	P	100110	10	1001110110
103	P	100111	13	1001011110110
104	P	101000	5	00001
105	P	101001	10	1001111010
106	P	101010	9	100111110
107	P	101011	12	000000111110
108	P	101100	6	001000
109	P	101101	11	10000010011
110	P	101110	10	0000001100
111	P	101111	11	10010111110
112	P	110000	5	10100
113	P	110001	11	10000010010
114	P	110010	10	1001010011
115	P	110011	12	100101111010
116	P	110100	6	100001
117	P	110101	11	10010101011
118	P	110110	10	1000001011
119	P	110111	12	000000110100
120	P	111000	5	10101
121	P	111001	10	1001111000
122	P	111010	10	1001010010
123	P	111011	12	100101101000
124	P	111100	5	00101
125	P	111101	10	0000001011
126	P	111110	8	10011100
127	P	111111	10	0000001110

[0060] In der Implementierung, die in Fig. 5 gezeigt ist, werden I-Frames unterschiedlich codiert als P-Frames, indem der Codierer eine Vorhersage verwendet, um die räumliche Abhängigkeit der codierten Blockparameter auszunutzen. Für jeden Makroblock beginnt der Encoder damit, die codierten Blockparameter für Chrominanz und Helligkeit zu erlangen, wie es in Schritt **106** gezeigt ist.

[0061] Als Nächstes berechnet der Codierer den Prädiktor der codierten Blockparameter für Hellig keit. In dieser bestimmten Implementierung verwendet der Codierer nur eine Vorhersage für die CBPY-Parameter. Jedoch könnte dieselbe Vorhersage auch verwendet werden, um die codierten Blockparameter für Chrominanz vorherzusagen. In dem Fall der Chrominanz wird die Vorhersage eher basierend auf 8×8 Pixel-Chrominanzblöcken in Nachbar-Makroblöcken berechnet als auf benachbarten 8×8 Pixel-Helligkeitsblöcken, welche in demselben Makroblock oder einem benachbarten Makroblock sein können. Weil jeder Makroblock vier Helligkeitsblöcke hat, können die Nachbarblöcke für einen gegebenen Helligkeitsblock von demselben oder einem Nachbar-Makroblock sein. Für die Vorhersage, die Chrominanzblöcke involviert, kommen die Nachbarblöcke von Nachbar-Makroblöcken.

[0062] Der Codierer führt eine räumliche Vorhersage auf codierte Blockparameter durch. Als Erstes schaut er bei den codierten Blockparametern für Nachbarblöcke nach, um zu ermitteln, ob sich, was wahrscheinlich ist, der Wert des Blockparameters von einem benachbarten Block zu dem aktuellen Block verändert. Wenn der Ort eines Blocks, der die kleinste Veränderung in dem codierten Blockparameter repräsentiert, identifiziert werden kann (d.h. der geringste räumliche Gradient in den codierten Blockparametern), dann wird an diesem Ort

der codierte Blockparameter für den Block als der Prädiktor verwendet. Andernfalls macht es keinen Unterschied, welcher Nachbar als Prädiktor gewählt wird und es wird lediglich irgendeiner ausgewählt. Ein bestimmtes Beispiel zum Auswählen des Prädiktors wird unterhalb mit Bezug auf die Fig. 6 bis Fig. 8 detaillierter beschrieben und illustriert.

[0063] Im nächsten Schritt **110** berechnet der Codierer einen vorhergesagten Wert für die codierten Blockparameter. Der vorhergesagte Wert stellt die Veränderung in dem codierten Blockparameter für den Prädiktor-Block und den aktuellen Block dar. Um den vorhergesagten Wert zu berechnen, führt der Codierer eine bitweise exklusive ODER-Funktion (XOR) auf dem vorhergesagten Wert und einem aktuellen Blockwert durch. Der resultierende Vektor, der als CBPCY XOR bezeichnet wird, wird anschließend einem Code variabler Länge von einer Huffman-Tabelle zugewiesen. Der Codierer schaut den Eintrag für CBPCY XOR in der Tabelle nach und findet den entsprechenden Code variabler Länge. Tabelle 2 unterhalb zeigt die verwendete VLC-Tabelle, um vorhergesagte CBPCY-Werte für I-Frames in der Implementierung zu codieren.

Tabelle 2: VLC-Tabelle für codierte Blockmuster der Chrominanz und der Helligkeit für I-Bilder

Index	CBPCY_XOR (1234) UV	Number of bits	Code
0	000000	1	1
1	000001	6	010111
2	000010	5	01001
3	000011	5	00101

Index	CBPCY_XOR (1234) UV	Number of bits	Code
4	000100	5	00110
5	000101	9	001000111
6	000110	7	0100000
7	000111	7	0010000
8	001000	5	00010
9	001001	9	001111100
10	001010	7	0111010
11	001011	7	0011101
12	001100	6	000010
13	001101	9	011101100
14	001110	8	01110111
15	001111	8	00000000
16	010000	5	00011
17	010001	9	010110111
18	010010	7	0101100
19	010011	7	0010011
20	010100	6	000001
21	010101	10	0101101000
22	010110	8	01000110
23	010111	8	00111111
24	011000	6	011110
25	011001	13	0011100010010
26	011010	9	010110101
27	011011	8	01000010
28	011100	7	0100010
29	011101	11	00111000101
30	011110	10	0100011110
31	011111	9	010000111
32	100000	4	0110
33	100001	9	000000011
34	100010	7	0011110
35	100011	6	011100
36	100100	7	0010010
37	100101	12	001110001000
38	100110	9	001000100
39	100111	9	001110000
40	101000	6	011111
41	101001	11	01000111110
42	101010	8	00111001
43	101011	9	010001110
44	101100	7	0000001
45	101101	11	00111000110

Index	CBPCY_XOR (1234) UV	Number of bits	Code
46	101110	9	010110110
47	101111	9	001000101
48	110000	6	010100
49	110001	11	01000111111
50	110010	9	001111101
51	110011	9	000011000
52	110100	7	0000111
53	110101	11	00111000111
54	110110	9	010000110
55	110111	9	000011001
56	111000	6	010101
57	111001	10	0111011011
58	111010	9	000000010
59	111011	9	001000110
60	111100	8	00001101
61	111101	13	0011100010011
62	111110	10	0111011010
63	111111	10	0101101001

[0064] Fig. 6 bis Fig. 8 stellen die räumliche Vorhersage detaillierter dar, die von dem Codierer durchgeführt wird. Fig. 6 ist ein Diagramm, das vier benachbarte Makroblöcke zeigt (oben links **120**, oben rechts **122**, unten links **124** und unten rechts **126**). Das folgende Beispiel konzentriert sich auf den Block unten rechts, welcher mit einem Kreis versehen ist. Jeder der Makroblöcke schließt vier 8×8-Pixel-Blöcke für Helligkeit ein, die mit Y1, Y2, Y3 und Y4 bezeichnet sind.

[0065] Als ein Beispiel kann man sich den Helligkeitsblock links oben Y1 für den Makroblock **126** vorstellen. Die Blöcke, die verwendet werden, um den Prädiktor zu berechnen, sind von einer gestrichelten Linie **128** umgeben. Der Block, auf dem das Interesse liegt, ist Y1 (als Block **130a** gekennzeichnet), und die Blöcke, die verwendet werden, um den Prädiktor zu berechnen, sind die benachbarten Blöcke, die als **132a**, **134a** und **136a** gekennzeichnet sind.

[0066] Um ein spezifisches Beispiel zu geben, zeigt Fig. 7 Werte der codierten Blockmusterparameter für jeden der Blöcke innerhalb der gestrichelten Linie aus Fig. 6. Die Bezugsnummern **130b**, **132b**, **134b** und **136b** entsprechen den Blöcken **130a**, **132a**, **134a** bzw. **136a** aus Fig. 6. Die räumlichen Gradienten der benachbarten codierten Blockparameter werden verwendet, um den Prädiktor auszuwählen. Im Wesentlichen wird der vertikale Gradient von den codierten Blockparametern der benachbarten Blöcke oben links und links (**136a**, **132a**, und umkreist dargestellt **140** in Fig. 7) berechnet. Der horizontale Gradient wird von den codierten Blockparametern der benachbarten Blöcke links oben und oben (**136a**, **130a**, und umkreist gezeigt **142** in Fig. 7) berechnet.

[0067] Fig. 8 ist ein Ablaufdiagramm, das die Schritte zum Finden des Prädiktors darstellt. Als Erstes findet der Codierer die vertikalen und horizontalen Gradienten. Jeder wird mit der exklusiven ODER-Funktion mit den codierten Blockparametern berechnet, die umkreist in Fig. 7 gezeigt werden (**140** ist der vertikale Gradient und **142** ist der horizontale Gradient). Als Nächstes vergleicht der Codierer die Gradientenwerte. Wenn die Gradienten nicht dieselben sind, wählt der Codierer den Prädiktor als den Wert aus, der dem Block in der Richtung mit dem geringeren Gradient zugewiesen ist. In dem Beispiel, das in Fig. 7 gezeigt ist, ist der vertikale Gradient null, während der horizontale Gradient 1 ist. Deshalb geht die Richtung des geringeren Gradienten nach oben. Als solches wird der Wert des codierten Blockparameters für Block **134a** als der Prädiktor verwendet, weil er in der Richtung "nach oben" relativ zu dem Block, der von Interesse ist, liegt.

[0068] Ob eine Vorhersage verwendet wird, um die codierten Blockparameter zu modifizieren, oder nicht, das Endergebnis ist ein einzelner Code variabler Länge, der alle codierten Blockparameter für den Makroblock dar-

stellt. Weil I- und P-Frames unterschiedlich in der Implementierung codiert werden, behandelt der Decodierer die Makroblöcke für dieselben Frames unterschiedlich. Für P-Frames verwendet der Decodierer die VLC-Tabelle 1, um den einzelnen Code variabler Länge nachzusehen und den entsprechenden Eintrag zu finden, der die kombinierten Parameter speichert, die die codierten Blockparameter für Helligkeit und Chrominanz darstellen. Für I-Frames verwendet der Decodierer die VLC-Tabelle 2, um den einzelnen Code variabler Länge nachzusehen und den entsprechenden Eintrag zu finden, der den kombinierten Parameter speichert, der die codierten Blockparameter für Helligkeit und Chrominanz darstellt. Sowohl bei I- als auch P-Frames verwendet das Texturdecodierungsmodul (Block **74** in Fig. 3) die codierten Blockparameter, um zu ermitteln, ob die Texturdaten für den entsprechenden Block decodiert werden müssen. Der Decodierer überspringt Texturdecodieren für Blöcke, die einen codierten Blockparameter von null haben.

[0069] In den Fällen, wo die codierten Blockparameter ebenso vorhergesagt werden, verwendet der Decodierer die vorher decodierten Blockparameter von den Nachbarblöcken, um den codierten Blockparameter für den aktuellen Block, der von Interesse ist, zu berechnen. Als Erstes berechnet der Decodierer den Ort des Prädiktorblocks, basierend auf den räumlichen Gradienten in derselben Art und Weise, wie der Codierer. Als Nächstes berechnet er den Wert des codierten Blockparameters für den aktuellen Block durch das Berechnen der exklusiven ODER-Funktion von dem decodierten Wert und dem codierten Blockparameter des Prädiktorblocks (der exklusive ODER-Operator hat die folgende Eigenschaft: X XOR Y = Z; Z XOR X = Y). Nach dieser umgekehrten Vorhersagestufe verwendet der Texturdecodierer anschließend den codierten Blockparameter, um zu ermitteln, ob das Decodieren der Textur für den Block übersprungen wird.

Kurzer Überblick über ein Computersystem

[0070] Fig. 9 und die folgende Diskussion sind gedacht, um eine kurze, allgemeine Beschreibung einer geeigneten Computerumgebung bereitzustellen, in der die Erfindung implementiert werden kann. Obwohl die Erfindung und Aspekte davon in einem Hardwaregerät implementiert werden können, sind die oben beschriebenen Encoder und Decoder in computerausführbaren Instruktionen, die in Programmmodulen organisiert sind, implementiert. Die Programmmodule schließen die Routinen, Programme, Objekte, Komponenten und Datenstrukturen ein, die die Aufgaben durchführen und die Datentypen, die oberhalb beschrieben wurden, implementieren.

[0071] Während Fig. 9 eine typische Konfiguration eines Desktop-Computers zeigt, kann die Erfindung in anderen Computersystemkonfigurationen implementiert werden, einschließlich tragbaren Geräten, Multiprozessorsystemen, Mikroprozessor-basierter oder programmierbarer Unterhaltungselektronik, Minicomputer, Mainframe-Computer und Ähnliches. Die Erfindung kann ebenso in verteilten Computerumgebungen verwendet werden, wo Aufgaben durch Remote-Verarbeitungsgeräte durchgeführt werden, die durch ein Datenübertragungsnetzwerk verbunden sind. In einer verteilten Computerumgebung können Programmmodule sowohl lokal als auch auf Remote-Datenspeichergeräten liegen.

[0072] Fig. 9 stellt ein Beispiel eines Computersystems dar, das als eine Arbeitsumgebung für die Erfindung dient. Das Computersystem schließt einen Personalcomputer **920** ein, einschließlich einer Prozessoreinheit **921**, einem Systemspeicher **922** und einem Systembus **923**, der verschiedene Systemkomponenten einschließlich des Systemspeichers mit der Prozessoreinheit **921** verbindet. Der Systembus kann irgendeinen von verschiedenen Busstrukturtypen umfassen, einschließlich eines Speicherbusses oder Speichercontrollers, eines Peripheriebusses, und eines lokalen Busses, der eine Busarchitektur, wie z.B. PCI, VESA, Microchannel (MCA), ISA und EISA, um ein paar zu nennen, verwendet. Der Systemspeicher schließt nur Lesespeicher (read only memory – ROM) **924** und Arbeitsspeicher (random access memory – RAM) **925** ein. Ein Basic-Input/Output-System (BIOS) **926**, das die Basisroutinen enthält, die helfen, Informationen zwischen Elementen in dem Personalcomputer **920** zu übertragen, z.B. während des Hochfahrens, ist in dem ROM **924** gespeichert. Der Personalcomputer **920** enthält des Weiteren ein Festplattenlaufwerk **927**, ein magnetisches Disklaufwerk **928**, z.B. um von/zu einer entfernbaren Disk **929** zu lesen oder zu schreiben, und ein optisches Disklaufwerk **930**, z.B. zum Lesen von einer CD-ROM-Disk **931** oder zum Lesen oder Schreiben von/zu eines anderen optischen Datenträgers. Das Festplattenlaufwerk **927**, magnetische Disklaufwerk **928** und optische Disklaufwerk **930** sind mit dem Systembus **923** über eine Festplattenlaufwerksschnittstelle **932**, eine Schnittstelle **933** für magnetische Disklaufwerke bzw. eine Schnittstelle **934** für optische Laufwerke verbunden. Die Laufwerke und ihre zugehörigen computerlesbaren Datenträger stellen nicht-flüchtigen Speicher für Daten, Datenstrukturen, computerausführbare Instruktionen (Programmcode, wie z.B. dynamisch verknüpfte Bibliotheken und ausführbare Dateien) etc., für den Personalcomputer **920** bereit. Obwohl die Beschreibung von computerlesbaren Datenträgern sich oberhalb auf eine Festplatte, eine entfernbare magnetische Disk und eine CD bezieht, können sie ebenso andere Typen von Datenträgern einschließen, die von einem Computer lesbar sind,

wie z.B. magnetische Kassetten, Flash-Memory-Karten, Digital-Video-Disks, Bernoulli-Kartuschen und Ähnliches.

[0073] Eine Anzahl von Programmmodulen kann in den Laufwerken und dem RAM **925** gespeichert werden, einschließlich eines Betriebssystems **935**, eines oder mehrerer Anwendungsprogramme **936**, anderer Programmmodule **937** und Programmdaten **938**. Ein Benutzer kann Befehle und Informationen in den Personalcomputer **920** durch eine Tastatur **940** und Zeigergerät, wie z.B. eine Maus **942**, eingeben. Andere Eingabegeräte (nicht gezeigt) können ein Mikrofon, Joystick, Gamepad, Satellitenschüssel, Scanner oder Ähnliches einschließen. Diese und andere Eingabegeräte sind oft mit der Prozessoreinheit **921** durch eine serielle Anschlussschnittstelle **946** verbunden, die mit dem Systembus gekoppelt ist, aber können auch durch andere Schnittstellen verbunden sein, wie z.B. einen Parallelanschluss, Gameport oder einen Universal-Serial-Bus (USB). Ein Monitor **947** oder anderer Typ an Anzeigegerät ist ebenso mit dem Systembus **923** über eine Schnittstelle, wie z.B. einen Anzeigecontroller oder Video-Adapter **948**, verbunden. Zusätzlich zu dem Monitor schließen Personalcomputer üblicherweise andere periphere Ausgabegeräte (nicht gezeigt) ein, wie z.B. Lautsprecher und Drucker.

[0074] Der Personalcomputer **920** kann in einer Netzwerkumgebung unter Verwendung logischer Verbindungen zu einem oder mehreren Remotecomputern, wie z.B. einem Remotecomputer **949**, arbeiten. Der Remotecomputer **949** kann ein Server, Router, Peer-Gerät oder anderer bekannter Netzwerkknoten sein und enthält üblicherweise viele oder alle der oben mit Bezug auf den Personalcomputer **920** beschriebenen Elemente, obwohl nur ein Datenspeichergerät **950** in Fig. 9 dargestellt worden ist. Die logischen Verbindungen, die in Fig. 9 gezeigt sind, schließen ein local area network (LAN) **951** und ein wide area network (WAN) **952** ein. Solche Netzwerkumgebungen sind in Büros, unternehmensweiten Computernetzwerken, Intranets und dem Internet alltäglich.

[0075] Wenn er in einer LAN-Netzwerkumgebung verwendet wird, ist der Personalcomputer **920** mit dem lokalen Netzwerk **951** durch eine Netzwerkschnittstelle oder -adapter **953** verbunden. Wenn er in einer WAN-Netzwerkumgebung verwendet wird, enthält der Personalcomputer **920** üblicherweise ein Modem **954** oder andere Mittel zum Herstellen von Datenübertragungen über das wide area network **952**, wie z.B. das Internet. Das Modem **954**, welches intern oder extern sein kann, ist mit dem Systembus **923** über eine serielle Anschlussschnittstelle **946** verbunden. In einer Netzwerkumgebung können Programmmodule, die relativ zum Personalcomputer **920** gezeigt sind, oder Teile davon, in dem Remote-Datenspeichergerät gespeichert sein. Die gezeigten Netzwerkverbindungen sind lediglich Beispiele, und andere Mittel zum Herstellen einer Datenübertragungsverbindung zwischen den Computern können verwendet werden.

Ergebnis

[0076] Während die Erfindung unter der Verwendung spezifischer Implementationen als ein Beispiel dargestellt worden ist, ist der Umfang der Erfindung nicht auf die spezifische Implementierung beschränkt. Zum Beispiel ist es möglich, eine räumliche Vorhersage sowohl für Chrominanz- als auch Helligkeitsblöcke unter Verwendung ähnlicher Techniken zu verwenden. Zusätzlich kann eine räumliche Vorhersage zum Codieren der codierten Blockparameter sowohl für Intra- als auch vorhergesagte Frames verwendet werden. Die Implementierung verwendet Huffman-Tabellen, um Codes variabler Länge zu generieren. Tatsächlich kann eine Vielfalt von Entropie-Codierungsverfahren verwendet werden, um einen Code variabler Länge für jeden kombinierten codierten Blockparameter zu generieren. Zum Beispiel können verschiedene Formen der Arithmetik und/oder von Laufstreckencodierung (run length encoding) verwendet werden. Jedes dieser Codierungsverfahren weist Eingangssignalen, die weniger häufig auftreten, längere Codes zu, während häufigeren Eingangssignalen kürzere Codes zugewiesen werden. Wie es oben erwähnt wurde, können die Codierungsverfahren zum Verbessern der Effizienz der Makroblock-Header auf Frame-basierte und objektbasierte Codierungsverfahren angewandt werden.

Patentansprüche

1. Verfahren zum Rekonstruieren eines oder mehrerer Videobilder in einer Videosequenz während des Videodecodierens in einem Computersystem, wobei das Verfahren charakterisiert ist durch:
für jeden Makroblock von vielen Makroblöcken:
das Empfangen eines Codes mit variabler Länge in einem Bitstream, wobei ein codiertes Blockmuster erste codierte Blocksmusterinformation für viele Helligkeitsblöcke (luminance blocks) des Makroblocks einschließt, wobei die codierten Blockmuster zweite codierte Blockmusterinformation für viele Chrominanzblöcke des Makroblocks einschließt, wobei der empfangene Code mit variabler Länge einen Codetabelleneintrag in einer Co-

detabelle kennzeichnet, wobei der Codetabelleneintrag einen kombinierten Parameter spezifiziert, der die erste codierte Blockmusterinformation und die zweite codierte Blockmusterinformation des codierten Blockmusters für den Makroblock repräsentiert, wobei das codierte Blockmuster angibt, welche der vielen Helligkeitsblöcke und der vielen Chrominanzblöcke entsprechende Transformationskoeffizientendaten (**86**) in dem Bitdatenstrom haben, und wobei der Makroblock einen Makroblocktyp aufweist;
das Abrufen des codierten Blocksmusters für den Makroblock, zumindest teilweise basierend auf dem empfangenen Code variabler Länge, wobei das Abrufen das Nachschlagen des Codetabelleneintrags in der Codetabelle einschließt, und wobei der Makroblocktyp für mindestens einen der vielen Makroblöcke, für die codierte Blockmuster abgerufen wurden, Intra ist; und
das Verwenden des codierten Blocksmusters für den Makroblock während der Rekonstruktion des einen oder der mehreren Videobilder.

2. Verfahren nach Anspruch 1, wobei jeder Makroblock der vielen Makroblöcke aus vier Helligkeitsblöcken und zwei Chrominanzblöcken besteht.

3. Verfahren nach Anspruch 1, wobei jeder Makroblock der vielen Makroblöcke ein 16×16-Array (16×16-Bereich) aus Pixeln einschließt, und wobei jeder Block der vielen Helligkeitsblöcke und der vielen Chrominanzblöcke ein 8×8-Array aus Pixeln einschließt.

4. Verfahren nach Anspruch 1, wobei das Abrufen des codierten Blockmusters des Weiteren zumindest teilweise auf decodierten Daten von einem oder mehreren anderen Makroblöcken basiert.

5. Verfahren nach Anspruch 4, wobei die decodierten Daten von dem einen oder den mehreren Makroblöcken decodierte codierte Blockmusterinformationen umfassen.

6. Verfahren nach Anspruch 1, wobei der empfangene Code variabler Länge unterschiedlich ist, abhängig davon, ob der Makroblocktyp des Makroblocks Intra ist.

7. Verfahren nach Anspruch 1, wobei das codierte Blockmuster für jeden Makroblock der vielen Makroblöcke aus sechs Bits besteht.

8. Verfahren nach Anspruch 7, wobei vier der sechs Bits angeben, welche Blöcke der vielen Helligkeitsblöcke entsprechend Transformationskoeffizientendaten in dem Bitdatenstrom aufweisen.

9. Verfahren zum Codieren eines oder mehrerer Videobilder einer Videosequenz in einem Computersystem, wobei das Verfahren charakterisiert ist durch:
das Empfangen eines oder mehrerer Videobilder einer Videosequenz; und
für jeden Makroblock von vielen Makroblöcken des einen oder der mehreren Videobilder:
Ermitteln eines Codes variabler Länge für ein codiertes Blockmuster des Makroblocks, wobei das codierte Blockmuster erste codierte Blockmusterinformation (**100**, **106** "CBPY") für viele Helligkeitsblöcke (luminance blocks) des Makroblocks einschließt, wobei das codierte Blockmuster zweite codierte Blockmusterinformation (**100**, **106** "CBPC") für viele Chrominanzblöcke des Makroblocks einschließt, wobei der Code variabler Länge einen Codetabelleneintrag in einer Codetabelle kennzeichnet, wobei der Codetabelleneintrag einen kombinierten Parameter spezifiziert, der die erste codierte Blockmusterinformation und die zweite codierte Blockmusterinformation des codierten Blockmusters für den Makroblock repräsentiert, wobei das codierte Blockmuster angibt, welche der vielen Helligkeitsblöcke und der vielen Chrominanzblöcke entsprechende Transformationskoeffizientendaten (**86**) in einem Bitdatenstrom haben, wobei der Makroblock einen Makroblocktyp aufweist, und wobei der Markoblocktyp für mindestens einen der vielen Makroblöcke, für die Codes variabler Länge für codierte Blockmuster ermittelt wurden, Intra ist; und
Ausgeben des Codes variabler Länge in den Bitdatenstrom, wobei ein Videodecoder, der den Code variabler Länge empfängt, das codierte Blockmuster für den Makroblock zumindest teilweise basierend auf dem Nachschlagen des Codetabelleneintrags in der Codetabelle ermittelt.

10. Verfahren nach Anspruch 9, wobei jeder Makroblock der vielen Makroblöcke aus vier Helligkeitsblöcken und zwei Chrominanzblöcken besteht.

11. Verfahren nach Anspruch 9, wobei jeder Makroblock der vielen Makroblöcke ein 16×16-Array (16×16-Bereich) aus Pixeln einschließt, und wobei jeder Block der vielen Helligkeitsblöcke und der vielen Chrominanzblöcke ein 8×8-Array aus Pixeln einschließt.

12. Verfahren nach Anspruch 9, wobei das Ermitteln des Codes variabler Länge des Weiteren zumindest teilweise auf Daten von einem oder mehreren anderen Markoblöcken basiert.

13. Verfahren nach Anspruch 12, wobei die Daten von dem einen oder mehreren anderen Makroblöcken codierte Blockmusterinformation umfasst.

14. Verfahren nach Anspruch 9, wobei der Code variabler Länge unterschiedlich ist, abhängig davon, ob der Makroblocktyp des Makroblocks Intra ist.

15. Verfahren nach Anspruch 9, wobei das codierte Blockmuster für jeden Makroblock der vielen Makroblöcke aus sechs Bits besteht.

16. Verfahren nach Anspruch 15, wobei vier von den sechs Bits angeben, welche Blöcke der vielen Helligkeitsblöcke entsprechend Transformationskoeffizientendaten in dem Bitdatenstrom aufweisen.

17. Ein computerlesbarer Datenträger, der computerausführbare Instruktionen speichert, die dazu dienen, das Computersystem zu veranlassen, das Verfahren von irgendeinem der Ansprüche 1 bis 16 durchzuführen.

Es folgen 4 Blatt Zeichnungen

Anhängende Zeichnungen

FIG. 1 (Stand der Technik)

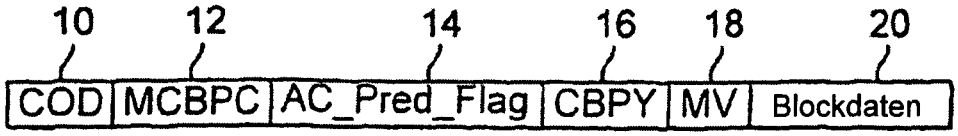


FIG. 4

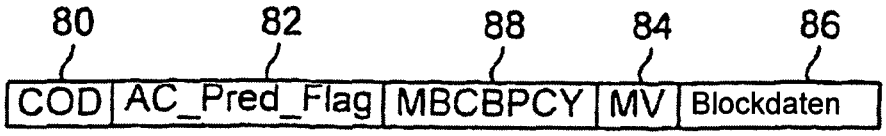


FIG. 5

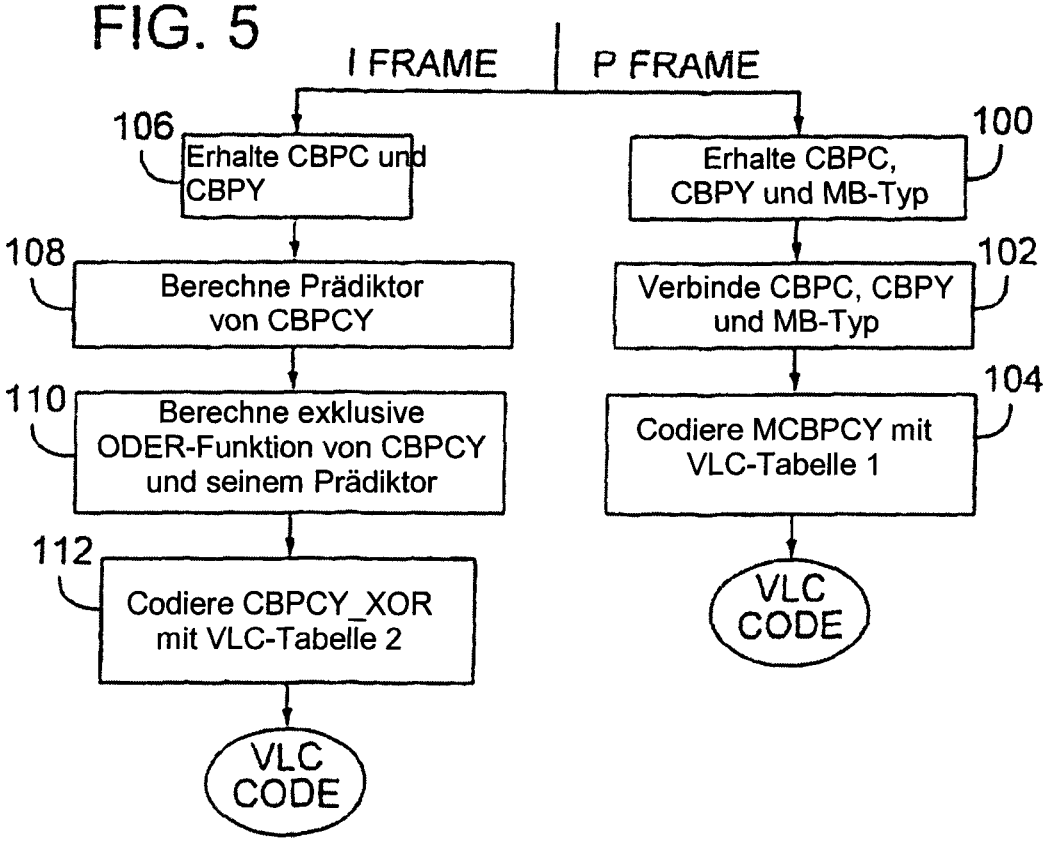


FIG. 2

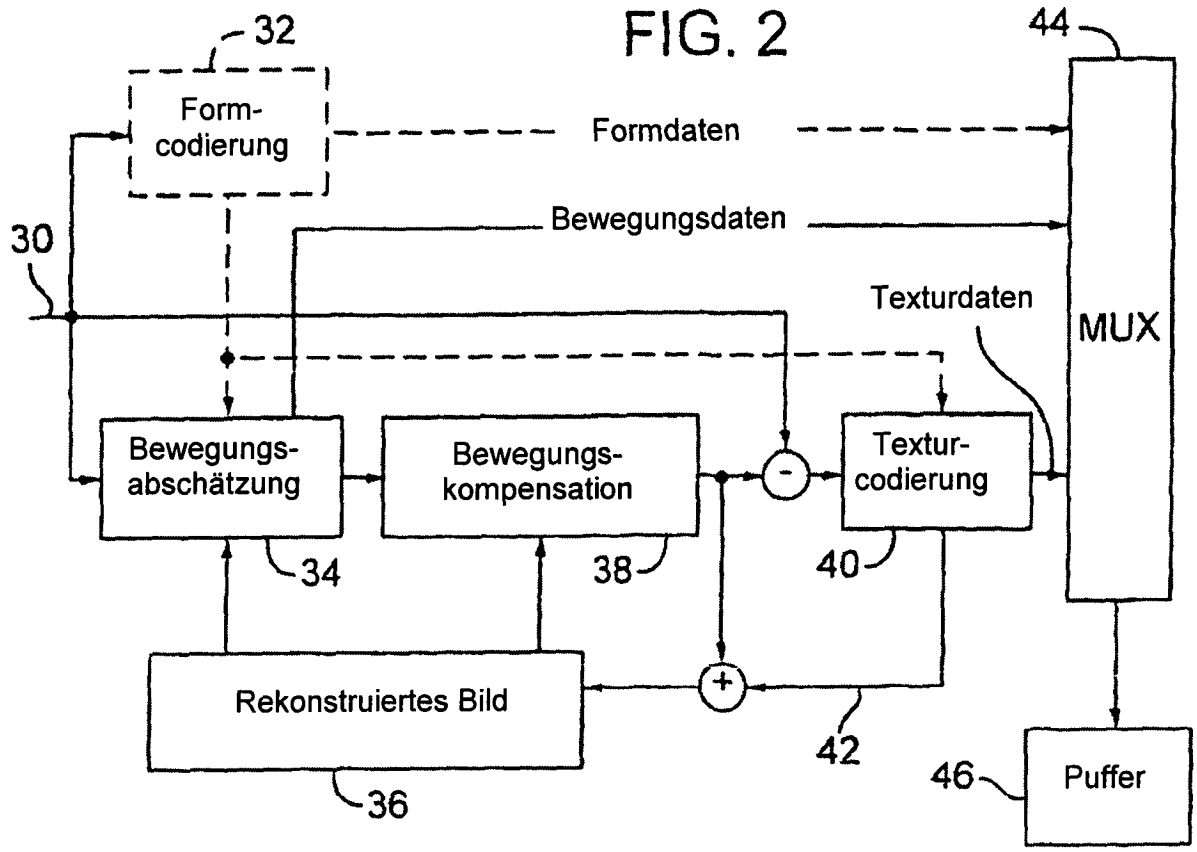

32
Form-
codierung
Formdaten
44
MUX
Bewegungsdaten
30
Texturdaten
Bewegungs-
abschätzung
Bewegungs-
kompensation
Textur-
codierung
34
38
40
Rekonstruiertes Bild
42
46
Puffer
36

FIG. 3

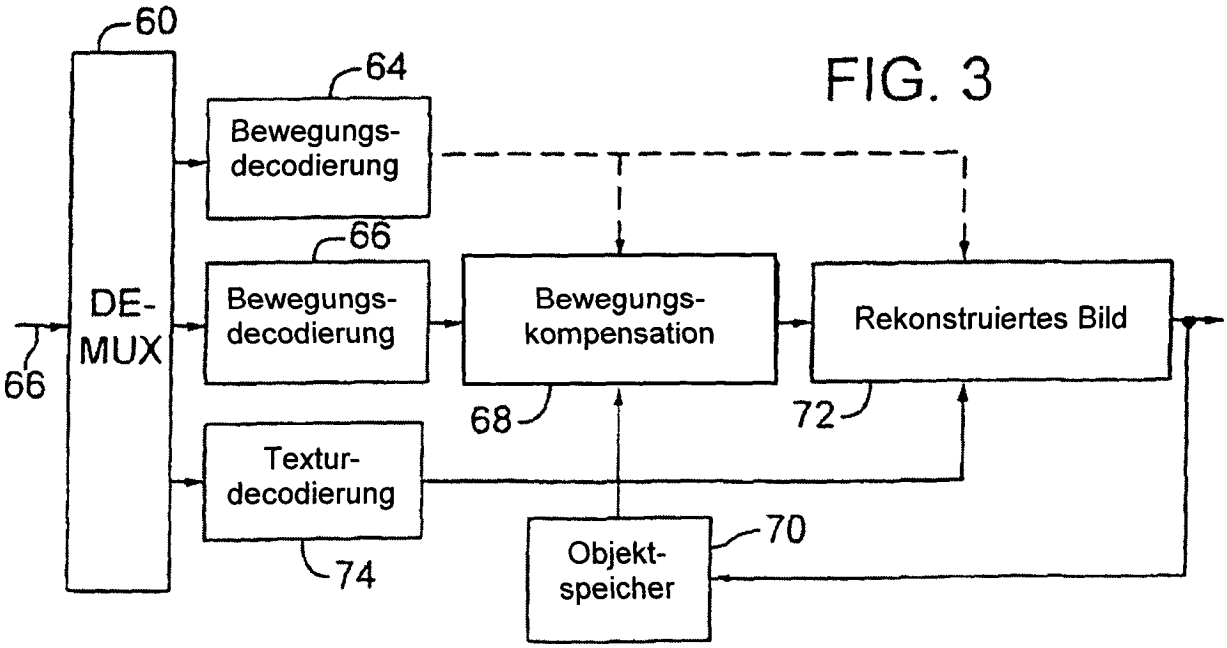

60
64
Bewegungs-
decodierung
66
DE-
MUX
Bewegungs-
decodierung
Bewegungs-
kompensation
Rekonstruiertes Bild
66
68
72
Textur-
decodierung
70
Objekt-
speicher
74

FIG. 6

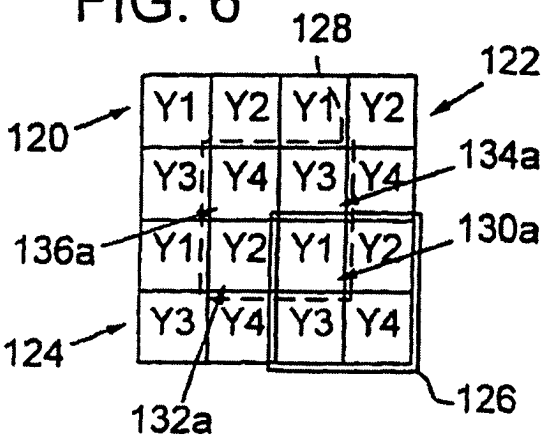


FIG. 7

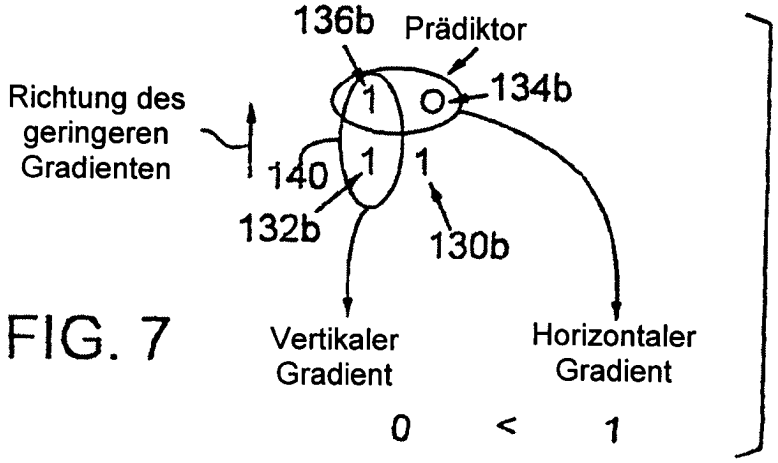


FIG. 8

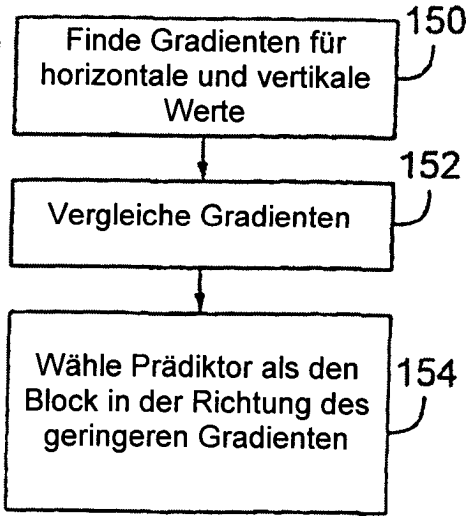
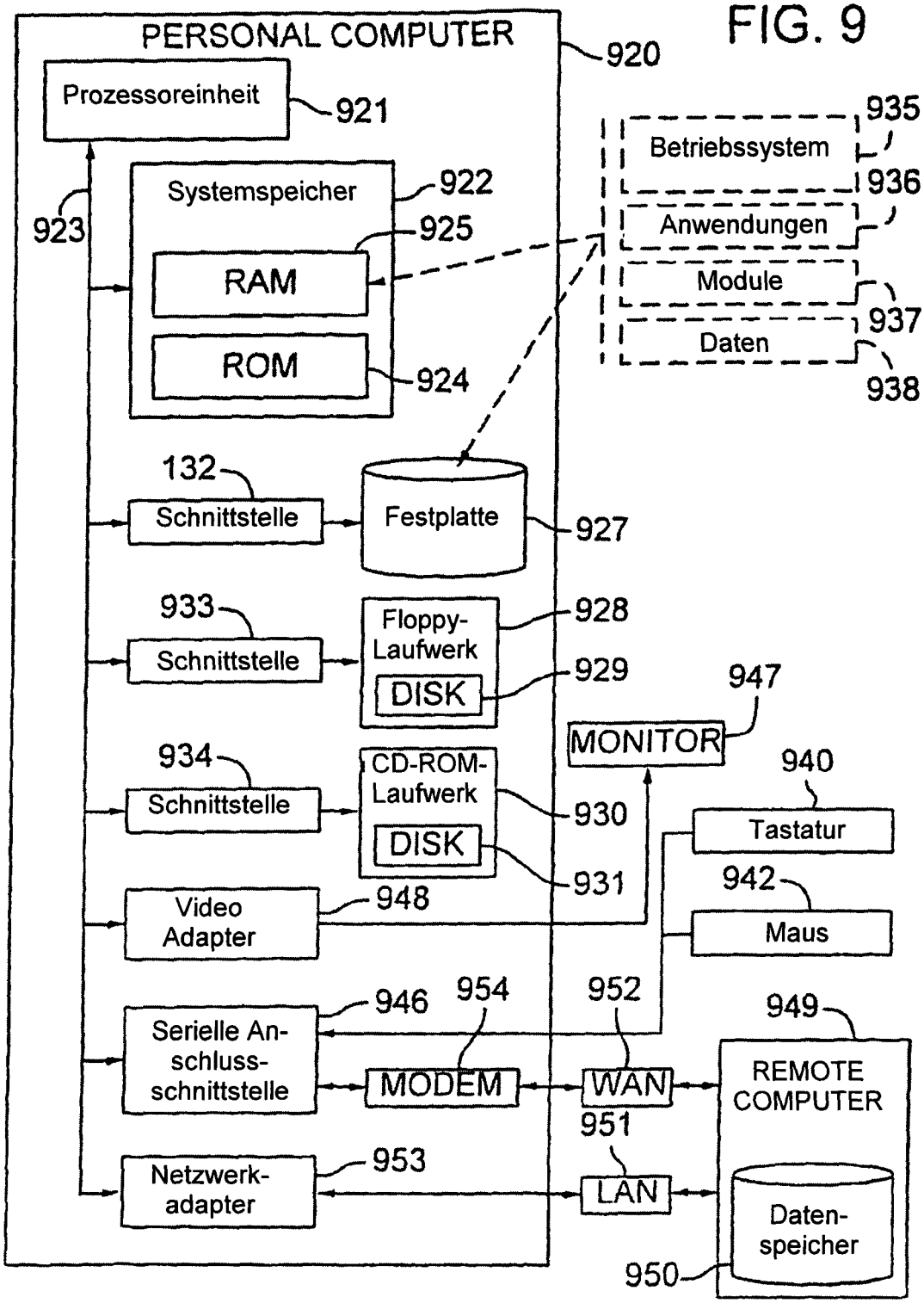


FIG. 9


PERSONAL COMPUTER
920
Prozessoreinheit
921
Systemspeicher
922
923
925
RAM
ROM
924
Betriebssystem
935
936
Anwendungen
Module
937
Daten
938
132
Schnittstelle
Festplatte
927
933
Schnittstelle
Floppy-Laufwerk
928
DISK
929
947
MONITOR
934
Schnittstelle
CD-ROM-Laufwerk
930
DISK
931
940
Tastatur
942
Maus
Video Adapter
948
946
954
952
949
Serielle An-schluss-schnittstelle
MODEM
WAN
951
REMOTE COMPUTER
953
Netzwerk-adapter
LAN
Daten-speicher
950