(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 374 064**

(51) Int. CI.:
G06T 1/00 (2006.01)
G06T 15/20 (2011.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Número de solicitud europea: **02258158 .1**
(96) Fecha de presentación: **27.11.2002**
(97) Número de publicación de la solicitud: **1321893**
(97) Fecha de publicación de la solicitud: **25.06.2003**

(54) Título: **ESTRUCTURA DE NODOS PARA REPRESENTAR OBJETOS TRIDIMENSIONALES USANDO IMÁGENES CON PROFUNDIDAD.**

(30) Prioridad:
27.11.2001 US 333167 P
08.03.2002 US 362545 P
01.05.2002 US 376563 P
12.07.2002 US 395304 P
04.11.2002 KR 2002067971

(45) Fecha de publicación de la mención BOPI:
13.02.2012

(45) Fecha de la publicación del folleto de la patente:
13.02.2012

(73) Titular/es:
SAMSUNG ELECTRONICS CO., LTD.
416, MAETAN-DONG, PALDAL-GU
SUWON-CITY, KYUNGKI-DO, KR

(72) Inventor/es:
Zhirkov, Alexander Olegovich;
Park, In-kyu;
Levkovich-Maslyuk, Leonid Ivanovich;
Ignatenko, Alexey Victorovich;
Han, Mahn-jin;
Bayakovski, Yuri Matveevich;
Konouchine, Anton y
Timasov, Dmitri Alexandrovich

(74) Agente: **Carpintero López, Mario**

DESCRIPCIÓN

Estructura de nodos para representar objetos tridimensionales usando imágenes con profundidad

La presente invención se refiere a una estructura de nodos para representar objetos tridimensionales (3D) basados en imágenes con profundidad y, más específicamente, a una estructura de nodos para representar objetos usando imágenes con información de profundidad.

Desde el comienzo de las investigaciones sobre los gráficos tridimensionales (3D), el fin último de los investigadores es sintetizar una escena gráfica realista, como una imagen real. Por lo tanto, se han llevado a cabo investigaciones sobre tecnologías tradicionales de representación, usando modelos poligonales y, como resultado, las tecnologías de modelización y representación se han desarrollado lo suficiente como para proporcionar entornos tridimensionales muy realistas. Sin embargo, el proceso para generar un modelo complicado necesita una gran cantidad de esfuerzos por parte de los expertos, y lleva mucho tiempo. Además, un entorno realista y complicado necesita una enorme cantidad de información y causa que se reduzca la eficacia en el almacenamiento y la transmisión.

Actualmente, los modelos poligonales se usan habitualmente para la representación de objetos tridimensionales en los gráficos por ordenador. Una forma arbitraria puede ser representada esencialmente por conjuntos de polígonos en color, es decir, triángulos. Algoritmos de software sumamente avanzados, y el desarrollo de hardware para gráficos, hacen posible visualizar objetos y escenas complejas como modelos poligonales de imágenes fijas y móviles considerablemente realistas.

Sin embargo, la búsqueda de representaciones tridimensionales alternativas ha sido muy activa durante la última década. Las principales razones para esto incluyen la dificultad de construir modelos poligonales para objetos del mundo real, así como la complejidad de representación y la calidad insatisfactoria para producir una escena fotográfica verdaderamente realista.

Las aplicaciones exigentes requieren enormes cantidades de polígonos; por ejemplo, el modelo detallado de un cuerpo humano contiene varios millones de triángulos, que no son fáciles de manipular. Aunque el progreso reciente en las técnicas de búsqueda de distancias, tales como el escáner de distancias por láser, nos permite adquirir datos densos de distancias con error tolerable, aún es muy caro y también muy difícil obtener un modelo poligonal completo sin fisuras del objeto entero. Por otra parte, los algoritmos de representación para obtener calidad fotográfica realista son complejos en términos de cálculo y, por tanto, están lejos de la representación en tiempo real.

Es un aspecto de esta invención proporcionar una estructura de nodos para representar objetos tridimensionales (3D) usando imágenes con profundidad, para gráficos y animación por ordenador, llamada representación basada en imágenes con profundidad (DIBR), que ha sido adoptado en la Extensión del Entorno de Animación (AFX) del estándar MPEG-4.

Según un aspecto de la invención, una estructura de nodo según la reivindicación 1 es un campo de profundidad en el cual se registra un valor de profundidad para cada píxel.

Según la presente invención, el tiempo de representación para nodos basados en imágenes es proporcional al número de píxeles en las imágenes de referencia y de salida pero, en general, no a la complejidad geométrica, como en el caso poligonal. Además, cuando la representación basada en imágenes se aplica a los objetos y escenas del mundo real, la representación con realismo fotográfico de una escena natural se hace posible sin el uso de millones de polígonos y de cálculos caros.

Los objetos y ventajas anteriores de la presente invención devendrán más evidentes al describir en detalle las realizaciones preferidas de la misma, con referencia a los dibujos adjuntos, en los cuales:

la FIG. 1 es un diagrama de ejemplos de IBR integrados en software actual de referencia;

la FIG. 2 es un diagrama de una estructura de octárbol y del orden de los hijos;

la FIG. 3 es un gráfico que muestra la razón de compresión del Octárbol;

la FIG. 4 es un diagrama de ejemplos de una Imagen de Profundidad en Capas (LDI): (a) muestra la proyección del objeto, donde las células oscuras (voxels) corresponden a los 1, y las células blancas a los 0, y (b) muestra una sección bidimensional en (x, profundidad);

la FIG. 5 es un diagrama que muestra el componente de color del modelo del “Ángel”, después de recomponer sus datos de color;

la FIG. 6 es un diagrama que muestra la invariancia ortogonal de la probabilidad de ocurrencia de los nodos; (a) muestra los nodos actual y padre originales, y (b) muestra los nodos actual y padre, girados alrededor del eje y en 90 grados;

las FIGs. 7, 8 y 9 son figuras de compresión geométrica para el procedimiento basado en el mejor PPM (Mapa Portátil de Píxeles);

la FIG. 10 es un diagrama que muestra dos formas de recomposición del campo de color del modelo de Textura Puntual del “Ángel” en una imagen bidimensional;

la FIG. 11 es un diagrama de ejemplos de geometría sin pérdidas y de compresión de color con pérdidas: (a) y (b) son, respectivamente, la versión original y la comprimida del modelo del “Ángel”, y (c) y (d) son, respectivamente, la versión original y comprimida del modelo “Morton256”;

la FIG. 12 es un diagrama que muestra un modelo BVO (Octárbol Volumétrico Binario) y un modelo TBVO (Octárbol Volumétrico Binario con Textura) del “Ángel”;

la FIG. 13 es un diagrama que muestra imágenes adicionales tomadas por cámaras adicionales en el TBVO: (a) es una imagen de índice de cámara, (b) es una primera imagen adicional y (c) es una segunda imagen adicional;

la FIG. 14 es un diagrama que muestra un ejemplo de grabación de un flujo de TBVO: (a) muestra una estructura de árbol TBVO. El color gris es un símbolo de textura “indefinida”. Cada color indica un índice de cámara, (b) muestra el orden de recorrido del octárbol en un nodo de BVO y los índices de cámara; (c) muestra el flujo de TBVO resultante, en el cual los cubos rellenados y el cubo del octárbol indican, respectivamente, los octetos de textura y los octetos de BVO;

las FIGs. 15, 17, 18 y 19 son diagramas que muestran, respectivamente, los resultados de la compresión del TBVO de “Ángel”, “Morton”, “Palmera512” y “Robot512”;

la FIG. 16 es un diagrama que muestra imágenes despojadas de los modelos “Ángel” y “Morton”;

la FIG. 20 es un diagrama de un ejemplo de la imagen de textura y del mapa de profundidad;

la FIG. 21 es un diagrama de un ejemplo de Imagen de Profundidad en Capas (LDI): (a) muestra la Proyección del

objeto y (b) muestra píxeles en capas;

la FIG. 22 es un diagrama de un ejemplo de Textura de Cuadro (BT), en el cual se usan seis TexturasSimples (pares de imagen y mapa de profundidad) para representar el modelo mostrado en el centro;

la FIG. 23 es un diagrama de un ejemplo de Textura de Cuadro Generalizada (GBT):

(a) muestra ubicaciones de cámara para el modelo ‘Palmera’, (b) muestra planos de imagen de referencia para el mismo modelo (se usan 21 TexturasSimples);

la FIG. 24 es un diagrama de un ejemplo que muestra la representación en Octárbol ilustrada en 2 dimensiones: (a) muestra una ‘nube de puntos’, (b) muestra los correspondientes mapas medios;

la FIG. 25 es seudocódigo para grabar el flujo de bits de TBVO;

la FIG. 26 es un diagrama que muestra la especificación de los nodos de DIBR;

la FIG. 27 es un diagrama del modelo de volumen de vista para la Imagen con Profundidad; (a) es una vista en perspectiva, (b) es una vista ortográfica;

la FIG. 28 es seudocódigo de representación basada en OpenGL de la TexturaSimple;

la FIG. 29 es un diagrama de un ejemplo que muestra la compresión de una imagen de referencia en TexturaSimple: (a) muestra la imagen de referencia original y (b) muestra la imagen de referencia modificada en un formato de JPEG;

la FIG. 30 es un diagrama de un ejemplo que muestra el resultado de representación del modelo “Morton” en distintos formatos; (a) está en un formato poligonal original, (b) está en un formato de Imagen con Profundidad y (c) está en un formato de Imagen de Octárbol;

la FIG. 31 es un diagrama de ejemplos de representación: (a) muestra el modelo de “Torre” escaneado en un formato de Imagen con Profundidad, (b) muestra el mismo modelo en un formato de Imagen de Octárbol (los datos del escáner se usaron sin eliminar el ruido y, por tanto, los puntos negros en la parte superior del modelo);

la FIG. 32 es un diagrama de ejemplos de representación del modelo “Palmera”: (a) muestra un formato poligonal original y (b) muestra el mismo modelo, pero en un formato de Imagen con Profundidad;

la FIG. 33 es un diagrama de un ejemplo de representación, que muestra una trama de la animación de “Dragón512” en

Imagen de Octárbol;

la FIG. 34 es un diagrama de un ejemplo de representación del modelo “Ángel512” en un formato de TexturaDePunto;

las FIGs. 35A y 35B son diagramas que muestran las relaciones de los respectivos nodos al representar un objeto en un formato de ImagenConProfundidad, con nodos de TexturaSimple y nodos de TexturaDePunto, respectivamente; y

la FIG. 36 es un diagrama que muestra la estructura del correspondiente nodo de ImagenOctárbol al representar un objeto con nodos de ImagenOctárbol.

1. ISO/IEC JTC 1/SC 29/WG 11 CODIFICACIÓN DE IMÁGENES EN MOVIMIENTO Y DE AUDIO

1. Introducción

En este documento, se informa del resultado del experimento central sobre Representación basada en Imágenes, AFX A8.3. Este experimento central es para tecnología de representación basada en imágenes, que usa texturas con información de profundidad. Además, en base a los experimentos después de la 57ª reunión del MPEG y los debates durante la reunión del Grupo Ad Hoc de AFX en Octubre, se presentan pocos cambios realizados en la especificación del nodo.

2. Resultados experimentales

2.1. Modelos de prueba

- Para objetos fijos
  - Nodo de ImagenConProfundidad con TexturaSimple
    - Perro
    - Tirannosaurus Rex (ImagenConProfundidad, usando alrededor de 20 cámaras)
    - Terrasque (un monstruo) (ImagenConProfundidad, alrededor de 20 cámaras)
    - ChumSungDae (ImagenConProfundidad, datos escaneados)
    - Palmera (ImagenConProfundidad, 20 cámaras)
  - Nodo de ImagenConProfundidad con Textura en Capas
    - Ángel
  - Nodo de ImagenConProfundidad con TexturaDePunto
    - Ángel
  - Nodo de ImagenOctárbol
    - Criatura
- Para objetos animados
  - Nodo de ImagenConProfundidad con TexturaSimple
    - Dragón
    - Dragón en entorno de escenario
  - Nodo de ImagenConProfundidad con Textura en Capas
    - No indicado
  - Nodo de ImagenOctárbol
    - Robot
    - Dragón en entorno de escenario
- Más datos (escaneados o modelados) se proporcionarán en el futuro.

2.2. Resultados de Pruebas

● Todos los nodos propuestos en Sydney se integran en el software de referencia blaxxun contact 4.3. Sin embargo, las fuentes no están aún cargadas en el servidor cvs.

● Los formatos animados de la IBR requieren sincronización entre múltiples ficheros de película, de modo tal que las imágenes en la misma trama clave de cada fichero de película deban darse a la vez. Sin embargo, el software de referencia actual no da soporte a esta capacidad de sincronización, que es posible en sistemas del estándar MPEG. Por lo tanto, actualmente, los formatos animados pueden visualizarse suponiendo que todos los datos de animación ya están en el fichero. Temporalmente, se usan ficheros de películas en un formato AVI para cada textura animada.

● Después de algunos experimentos con texturas en capas, nos convencimos de que el nodo de Textura en Capas no es eficaz. Este nodo fue propuesto para la Imagen con Profundidad en Capas. Sin embargo, también está el nodo de TexturaDePunto que puede darle soporte. Por lo tanto, proponemos eliminar el nodo de Textura en Capas de la especificación del nodo. La FIG. 1 muestra ejemplos de la IBR integrados en el software de referencia actual.

3. Actualizaciones en la Especificación del Nodo de IBR

La conclusión de la reunión de Sydney sobre la propuesta de la IBR fue tener un flujo de IBR que contiene imágenes e información de cámara, y el nodo de la IBR solamente tendrá un enlace (URL) con él. Sin embargo, durante la reunión de AhG en Rennes, el resultado del debate sobre la IBR fue tener imágenes e información de cámara tanto en los nodos de IBR como en el flujo. Así, la siguiente es la especificación actualizada de nodo para los nodos de IBR. Los requisitos para el flujo de IBR se dan en la sección que explica el campo URL.

Descodificador (Flujos de bits) – Especificación de nodo

ImagenConProfundidad {

campo	SFVec3f	posición	0 0 10
campo	SFRotation	orientación	0 0 1 0
campo	SFVec2f	campoDeVisión	0,785389 0,785398
campo	SFFloat	PlanoCercano	10
campo	SFFloat	PlanoLejano	100
campo	SFBool	ortogonal	FALSO
campo	SFNode	diTextura	NULO
campo	SFString	UrlImagenConProfundidad	“”

}

El nodo ImagenConProfundidad define una única textura de IBR. Cuando múltiples nodos de ImagenConProfundidad se relacionan entre sí, se procesan como un grupo y, por tanto, deberían colocarse debajo del mismo nodo de Transformada.

El campo diTextura especifica la textura con profundidad, que se transformará en la región definida en el nodo de ImagenConProfundidad. Será uno de los diversos tipos de texturas de imagen con profundidad (TexturaSimple o TexturaDePunto).

Los campos de posición y orientación especifican la ubicación relativa del punto de vista de la textura de la IBR en el sistema de coordenadas local. La posición es relativa al origen (0, 0,0) del sistema de coordenadas, mientras que la orientación especifica una rotación relativa a la orientación por omisión. En la posición y orientación por omisión, el espectador está sobre el eje Z, mirando hacia abajo por el eje Z, hacia el origen, con los X positivos a la derecha y los Y positivos hacia arriba. Sin embargo, la jerarquía de transformación afecta a la posición y orientación finales del punto de vista.

El campo campoDeVisión especifica un ángulo de visión desde el punto de vista de la cámara, definido por campos de posición y orientación. El primer valor indica el ángulo con el lado horizontal y el segundo valor indica el ángulo con el lado vertical. Los valores por omisión son de 45 grados en radianes. Sin embargo, cuando el campo ortogonal se establece como VERDADERO, el campo campoDeVisión indica el ancho y la altura del plano cercano y del plano lejano.

Los campos del PlanoCercano y del PlanoLejano especifican las distancias desde el punto de vista hasta el plano cercano y el plano lejano del área de visibilidad. La textura y los datos de profundidad muestran el área encerrada por el plano

cercano, el plano lejano y el campoDeVisión. Los datos de profundidad están normalizados para la distancia desde el PlanoCercano hasta el PlanoLejano.

El campo ortogonal especifica el tipo de vista de la textura de la IBR. Cuando se fija como VERDADERO, la textura de la IBR se basa en la vista ortogonal. En caso contrario, la textura de la IBR se basa en la vista en perspectiva.

El campo UrlImagenConProfundidad especifica la dirección del flujo de imágenes con profundidad, que puede contener optativamente el siguiente contenido.

* posición

* orientación

* campoDeVisión

* PlanoCercano

* PlanoLejano

* ortogonal

* diTextura (TexturaSimple o TexturaDePunto)

* cabecera de 1 octeto para los indicadores de activo / inactivo de los campos anteriores

TexturaSimple {

campo SFNode textura NULO

campo SFNode profundidad NULO

}

El nodo de TexturaSimple define una capa única de textura de IBR.

El campo de textura especifica la imagen plana que contiene color para cada píxel. Será uno de los diversos tipos de nodos de textura (TexturaDeImagen, TexturaDePelícula o TexturaDePíxel).

El campo de profundidad especifica la profundidad para cada píxel en el campo de textura. El tamaño del mapa de profundidades será el mismo tamaño que el de la imagen o película en el campo de textura. Será uno de los diversos tipos de nodos de textura (TexturaDeImagen, TexturaDePelícula o TexturaDePíxel). Si el nodo de profundidad es NULL, o si el campo de profundidad no está especificado, se usará el canal alfa en el campo de textura como el mapa de profundidad.

TexturaDePunto {

campo SFInt32 ancho 256

campo SFInt32 alto 256

campo MFInt32 profundidad []

campo MFColor color []

}

El nodo de TexturaDePunto define múltiples capas de puntos de IBR.

Los campos de ancho y alto especifican el ancho y el alto de la textura.

El campo de profundidad especifica múltiples profundidades de cada punto (en coordenadas normalizadas) en el plano proyectado en el orden de la trayectoria, que comienza desde el punto en la esquina inferior izquierda y avanza a la derecha para acabar la línea horizontal antes de avanzar a la línea superior. Para cada punto, se almacena primero el número de profundidades (píxeles), y ese número de valores de profundidad vendrá a continuación.

El campo de color especifica el color del píxel actual. El orden será el mismo que el del campo de profundidad, excepto en que no se incluye el número de profundidades (píxeles) para cada punto.

ImagenOctárbol {

campo	SFInt32	resoluciónoctárbol	256
campo	SFString	octárbol	“”
campo	MFNode	imágenesoctárbol	[]
campo	SFString	URLoctárbol	“”

}

El nodo ImagenOctárbol define una estructura de octárbol y sus texturas proyectadas. El tamaño del cubo circundante del octárbol total es 1 x 1 x 1, y el centro del cubo del octárbol será el origen (0, 0, 0) del sistema de coordenadas locales.

El campo resoluciónoctárbol especifica el máximo número de hojas del octárbol a lo largo de un lado del cubo circundante. El nivel del octárbol puede determinarse a partir de resoluciónoctárbol, usando la siguiente ecuación: niveloctárbol = ent(log2(resoluciónoctárbol-1))+1)

El campo octárbol especifica un conjunto de nodos internos del octárbol. Cada nodo interno está representado por un octeto. Un 1 en el i-ésimo bit de este octeto significa que existen los nodos hijos para el i-ésimo hijo de ese nodo interno, mientras que 0 significa que no existen. El orden de los nodos internos del octárbol será el orden de amplitud del primer recorrido del octárbol. El orden de ocho hijos de un nodo interno se muestra en la FIG. 2.

El campo imágenesoctárbol especifica un conjunto de nodos de ImagenConProfundidad con TexturaSimple para el campo diTextura. Sin embargo, no se usan el campo PlanoCercano y PlanoLejano del nodo ImagenConProfundidad y el campo profundidad en el nodo de Textura Simple.

El campo URLoctárbol especifica la dirección del flujo de Imagenoctárbol con el siguiente contenido.

* cabecera para indicadores
* resoluciónoctárbol
* octárbol
* imágenesoctárbol (Nodos múltiples de ImágenConProfundidad)

■ PlanoCercano no usado

■ PlanoLejano no usado

■ diTextura -> TexturaSimple sin profundidad

II. ISO/IEC JTC 1 / SC 29 / WG 11 CODIFICACIÓN DE IMÁGENES EN MOVIMIENTO Y DE AUDIO

1. Introducción

En este documento, se informa sobre el resultado del experimento central sobre la Representación basada en Imágenes con Profundidad (DIBR), AFX A8.3. Este experimento central es para los nodos de representación basada en imágenes con profundidad, que usan texturas con información de profundidad. Los nodos han sido aceptados e incluidos en una propuesta para el Borrador del Comité durante la reunión de Pattaya. Sin embargo, el flujo de esta información a través del campo URLoctárbol del nodo ImagenOctárbol y el campo URLImagenConProfundidad del nodo ImagenConProfundidad aún permanecía en marcha. Este documento describe el formato del flujo a enlazar por estos campos de URL. El formato del flujo incluye la compresión del campo octárbol del nodo ImagenOctárbol y los campos de profundidad / color del nodo de TexturaDePunto.

2. Formato de flujo para URLoctárbol

2.1. Formato del flujo

El nodo ImagenOctárbol incluye el campo URLoctárbol, que especifica la dirección del flujo de ImagenOctárbol. Este flujo puede contener, optativamente, el siguiente contenido.

* cabecera para indicadores
* resoluciónoctárbol

* octárbol

* imágenesoctárbol (Múltiples nodos de ImagenConProfundidad)

■ PlanoCercano no usado

■ PlanoLejano no usado

■ diTextura ->- TexturaSimple sin profundidad

El campo octárbol especifica un conjunto de nodos internos de octárbol. Cada nodo interno está representado por un octeto. Un 1 en el i-ésimo bit de este octeto significa que existen nodos hijos para el i-ésimo hijo de ese nodo interno, mientras que un 0 significa que no existen. El orden de los nodos internos de octárbol será el orden de recorrido, de amplitud primero, del octárbol. El orden de ocho hijos de un nodo interno se muestra en la FIG. 2.

El campo octárbol del nodo ImagenOctárbol está en un formato compacto. Sin embargo, este campo puede comprimirse adicionalmente a fin de tener un flujo eficaz. La siguiente sección describe el esquema de compresión para el campo octárbol del nodo ImagenOctárbol.

2.2. Esquema de compresión para el campo octárbol

En la representación del octárbol de la DIBR, los datos consisten en el campo de octárbol, que representa el componente de geometría. El octárbol es un conjunto de puntos en el cubo circundante, que representan completamente a la superficie del objeto.

La reconstrucción no idéntica de la geometría a partir de la representación comprimida lleva a efectos visuales sumamente notables. Por tanto, la geometría debe comprimirse sin pérdida de información.

2.2.1. Compresión de octárbol

Para la compresión del campo octárbol, representado en forma de octárbol de recorrido por profundidad primero, desarrollamos un procedimiento de compresión sin pérdidas, usando algunas ideas del enfoque PPM (Predicción por Coincidencia Parcial). La idea principal que usamos es la “predicción” (es decir, la estimación de probabilidad) del próximo símbolo por varios símbolos previos, que se llaman el “contexto”. Para cada contexto, existe una tabla de probabilidad, que contiene la probabilidad estimada de ocurrencia de cada símbolo en este contexto. Esto se usa en combinación con un codificador aritmético llamado codificador de distancias.

Las dos características principales del procedimiento son:

1. uso de un nodo padre como un contexto para el nodo hijo;

2. uso de una hipótesis de ‘invariancia ortogonal” para reducir el número de contextos.

La segunda idea se basa en la observación de que la ‘probabilidad de transición’ para pares de nodos ‘padre-hijo’ es habitualmente invariante por transformadas ortogonales (rotación y simetría). Esta hipótesis se ilustra en el Apéndice 1. Esta hipótesis nos permite usar un contexto más complejo sin tener demasiadas tablas de probabilidad. Esto, a su vez, nos permitió lograr resultados bastante buenos en términos de volumen y velocidad, porque cuantos más contextos se usan, más precisa es la estimación de probabilidad y, por tanto, más compacto es el código.

La codificación es el proceso de construir y actualizar la tabla probabilística según el modelo de contexto. En el procedimiento propuesto, el contexto se modela como la jerarquía de padre-hijo en la estructura de octárbol. En primer lugar, definimos el Símbolo como un nodo de octetos cuyos bits indican la ocupación del subcubo después de la subdivisión interna. Por lo tanto, cada nodo en el octárbol puede ser un símbolo y su valor numérico estará entre 0 y 255. La tabla probabilística (PT) contiene 256 valores enteros. El valor de la i-ésima variable ($0 \leq i \leq 255$), dividido entre la suma de todas las variables, es igual a la frecuencia (estimación de probabilidad) de la ocurrencia del i-ésimo símbolo. La Tabla de Contextos Probabilísticos (PCT) es un conjunto de las PT. La probabilidad de un símbolo está determinada a partir de una y sólo una de las PT. El número de la PT específica depende del contexto. Un ejemplo de PCT se muestra en la Tabla 1.

Tabla 1. Componente de una Tabla de Contextos Probabilísticos (PCT)

Identificador de las PT	0	1	...	255	Descripción del contexto
0	$P_{0,0}$	$P_{0,1}$	...	$P_{0,255}$	Contexto-0: independiente del contexto
1..27 (27)	$P_{i,0}$	$P_{i,1}$	...	$P_{i,255}$	Contexto-1: Símbolo Padre
28...243 (27*8)	$P_{j,0}$	$P_{j,1}$	...	$P_{j,255}$	Contexto-2: Símbolo Padre y Símbolo de Nodo

El codificador funciona de la siguiente manera. Primero usa el modelo de contexto-0 (es decir, una única PT para todos los símbolos, a partir de la distribución uniforme, y actualizando la PT después de cada nuevo símbolo codificado). El árbol se recorre en el orden de profundidad primero. Cuando se han reunido suficientes estadísticas (el valor empíricamente hallado es de 512 símbolos codificados), el codificador conmuta al modelo de contexto-1. Tiene 27 contextos, que se especifican según lo siguiente.

Consideremos un conjunto de 32 transformadas ortogonales fijas, que incluyen simetrías y rotaciones en 90 grados alrededor de los ejes de coordenadas (véase el Anexo 2). Entonces, podemos categorizar los símbolos según el patrón de relleno de sus subcubos. En nuestro procedimiento, habrá 27 conjuntos de símbolos, aquí llamados grupos, con la siguiente propiedad: 2 símbolos están conectados por una de estas transformadas fijas si y sólo si pertenecen al mismo grupo.

En la notación de octetos, los grupos se representan con 27 conjuntos de números (véase el Anexo 2). Suponemos que la tabla de probabilidades depende no del nodo padre en sí (en cuyo caso, habría habido 256 tablas), sino solamente del grupo (indicado como SímboloPadre en la FIG. 2) al cual pertenece el nodo padre (por tanto, 27 tablas).

En el momento de la conmutación, las PT para todos los contextos se fijan en copias de la PT de contexto-0. Luego, cada una de las 27 PT se actualiza cuando se usa para codificar.

Después de que 2.048 (otro valor heurístico) símbolos están codificados en el modelo de contexto-1, conmutamos al modelo de contexto-2, que usa los pares (SímboloPadre, SímboloNodo) como contextos. El SímboloNodo es sencillamente la posición del nodo actual en el nodo padre. Luego, tenemos 27*8 contextos para el modelo de contexto-2. En el momento de conmutar a ese modelo, las PT obtenidas para cada contexto se usan para cada nodo 'dentro' de este contexto, y a partir de este momento se actualizan independientemente.

Con algo más de detalle técnico, la codificación para los modelos de contexto-1 y contexto-2 se desarrolla de la siguiente manera. Para el contexto del símbolo actual (es decir, el nodo padre), se determina su grupo. Esto se hace por búsqueda en tabla (el análisis geométrico se llevó a cabo en la etapa del desarrollo del programa). Luego, aplicamos una transformada ortogonal que lleva nuestro contexto a un elemento "estándar" (seleccionado arbitrariamente de una vez y para siempre) del grupo al que pertenece. La misma transformada se aplica al símbolo en sí (estas operaciones también se implementan como búsqueda en una tabla, por supuesto – todos los cálculos para todas las posibles combinaciones se hicieron por anticipado). Efectivamente, esto es el cálculo de la posición correcta del símbolo actual en la tabla de probabilidades para el grupo que contiene su contexto. Luego la correspondiente probabilidad se suministra al CodificadorDeDistancias.

En breve, dado un símbolo padre y una posición de subnodo, se determina el IdentificadorDeContexto, que identifica el Identificador del grupo y la posición de la PT en la PCT. La distribución de probabilidades en la PT y el IdentificadorDeContexto se suministran a un codificador de distancias. Después de la codificación, la PCT se actualiza para usarse en la próxima codificación. Obsérvese que el codificador de distancias es una variación de la codificación aritmética, que efectúa la renormalización en octetos, en lugar de bits, funcionando así dos veces más rápido, y con una compresión peor en un 0,01% que una implementación estándar de la codificación aritmética.

El proceso de descodificación es esencialmente una inversión del proceso de codificación. Esto es un procedimiento absolutamente estándar que no requiere ser descrito, ya que usa exactamente los mismos procedimientos de determinación de contextos, de actualización de probabilidades, etc.

2.3. Resultados de pruebas

La FIG. 3 es una tabla para la comparación de nuestro enfoque, para modelos tanto fijos como animados (las ordenadas indican la razón de compresión). La razón de compresión del octárbol varía alrededor de entre 1,5 y 2 veces, en comparación con el tamaño original del octárbol, y supera en prestaciones las compresiones sin pérdida de propósito general (basadas en Lempel-Ziv, como el programa RAR) en hasta un 30%.

3. Formato de flujo para URLImagenConProfundidad

3.1. Formato del flujo

El nodo de ImagenConProfundidad incluye el campo URLImagenConProfundidad, que especifica la dirección del flujo de imágenes con profundidad. Este flujo puede contener optativamente el siguiente contenido.

* cabecera de 1 octeto para los indicadores de activado / desactivado de los campos a continuación

* posición

* orientación

* campoDeVisión

* PlanoCercano

* PlanoLejano

* ortogonal

* diTextura (TexturaSencilla o TexturaDePunto)

La definición del nodo de TexturaDePunto, que puede usarse en el campo diTextura del nodo ImagenConProfundidad, es la siguiente.

TexturaDePunto {

campo	SFInt32	ancho	256
campo	SFInt32	altura	256
campo	MFInt32	profundidad	[]
campo	MFColor	color	[]

}

El nodo TexturaDePunto define múltiples capas de puntos de IBR. Los campos de ancho y de altura especifican el ancho y la altura de la textura. El campo de profundidad especifica múltiples profundidades de cada punto (en coordenadas normalizadas) en el plano proyectado en el orden de recorrido, que comienza desde el punto en la esquina izquierda inferior y avanza hacia la derecha para acabar la línea horizontal antes de avanzar a la línea superior. Para cada punto, el número de profundidades (píxeles) se almacena primero, y ese número de valores de profundidad vendrán luego. El campo color especifica el color del píxel actual, El orden será el mismo que el campo de profundidad, excepto en que no se incluye ese número de profundidades (píxeles) para cada punto.

Los campos de profundidad y color de la TexturaDePunto están en un formato en bruto, y el tamaño de estos campos, muy probablemente, será muy grande. Por lo tanto, estos campos deben comprimirse a fin de tener un flujo eficaz. La siguiente sección describe el esquema de compresión para los campos del nodo TexturaDePunto.

3.2. Esquema de compresión para TexturaDePunto

3.2.1. Compresión del campo profundidad

El campo profundidad del nodo TexturaDePunto es sencillamente un conjunto de puntos en un ‘cubo circundante discretizado’. Suponemos que el plano inferior es el plano de proyección. Dadas las rejillas de dimensión m*n*l para un modelo, siendo puntos los centros de las células (en el caso del octárbol, los llamamos voxels) de esta rejilla, podemos considerar los voxels ocupados como los 1 y los voxels vacíos como los 0. El conjunto resultante de bits (m*n*l bits) se organiza luego en un flujo de octetos. Esto se hace recorriendo los voxels en la dirección de profundidad (ortogonal al plano de proyección), en capas de profundidad 8, y en el orden usual (“por columnas”) en el plano de proyección (rellenando, si es necesario, la última capa de octetos con ceros en el caso de que la dimensión de profundidad no sea un múltiplo de 8). Así, podemos pensar en nuestro conjunto de puntos como una pila de imágenes en escalas de grises de 8 bits (variante: imágenes de 16 bits). La correspondencia de voxels y bits se ilustra en la FIG. 4 (a).

Por ejemplo, en la FIG. 4 (b), los cuadrados negros corresponden a los puntos en el objeto. El plano horizontal es el plano de proyección. Consideremos la ‘tajada’ de la altura 16 (su límite superior se muestra en línea gruesa). Interpretemos las ‘columnas’ como octetos. Es decir, una columna por encima del punto marcado en la figura representa a la pila de 2

octetos, con valores 18 y 1 (o un entero 274 sin signo de 16 bits). Si aplicamos los mejores procedimientos disponibles de compresión basados en PPM a la unión de octetos obtenida de esta manera, se obtienen resultados bastante buenos. Sin embargo, si aquí se aplica directamente un sencillo procedimiento de contexto-1 (no puede usarse aquí ningún contexto de invariancia ortogonal, o jerárquico, por supuesto), esto da como resultado un grado levemente menor de compresión. A continuación damos una tabla de volúmenes requeridos para distintos tipos de representaciones de geometría de LDI: BVOC, la formación anterior de octetos comprimidos por el mejor compresor de PPM, y la misma formación comprimida por nuestro compresor usado actualmente (cifras en Koctetos).

Modelo	Representación por BVOC de la geometría	Mejor compresión de PPM de la formación de octetos	Compresión sencilla de contexto-1 de la formación de octetos
“Ángel”	31,4	27,5	32
“Morton”	23,4	23,3	30,5
“Saltamontes”	16,8	17,0	19,7

3.2.2. Compresión del campo color

El campo color del nodo TexturaDePunto es un conjunto de colores atribuidos a puntos del objeto. A diferencia del caso del octárbol, el campo color está en correspondencia de uno a uno con el campo profundidad. La idea es representar información de color como una única imagen, que podría ser comprimida por una de las técnicas con pérdida conocidas. La cardinalidad de esta imagen es mucho más pequeña que la de las imágenes de referencia en el octárbol o en el caso de la ImagenConProfundidad, y es una motivación esencial para tal enfoque. La imagen puede obtenerse recorriendo los puntos de profundidad en tal o cual orden natural.

Consideremos primero el orden de escaneo dictado por nuestro formato original de almacenamiento para LDI (TexturaDePunto) – escaneo por ‘profundidad-primero’ de la geometría. Los multipíxeles se recorren en el orden natural por el plano de proyección, como si fueran píxeles simples, y los puntos dentro del mismo multipíxel de recorren en la dirección de la profundidad. Este orden de escaneo produce una formación unidimensional de colores (1er multipíxel no nulo, 2º multipíxel no nulo, etc.). En cuanto se conoce la profundidad, los colores de los puntos pueden reconstruirse sucesivamente a partir de esta formación. Para hacer que los procedimientos de compresión de imágenes sean aplicables, debemos asociar unívocamente esta larga cadena a la formación bidimensional. Esto puede hacerse de muchas maneras.

El enfoque usado en las pruebas más adelante es el denominado “escaneo por bloques”, cuando la cadena de colores se dispone en bloques de 8*8, y se disponen esos bloques en orden por columnas (“escaneo por bloques”). La imagen resultante se muestra en la FIG. 5.

La compresión de esta imagen fue llevada a cabo por varios procedimientos, que incluyen el estándar JPEG. Resulta que al menos para este tipo de escaneo de colores, se obtienen resultados mucho mejores al usar el procedimiento de compresión de textura descrito en [5]. Este procedimiento se basa en la paletización local adaptable de cada bloque de 8*8. Tiene dos modalidades: compresión por 8 y por 12 (en comparación con el formato BMP ‘en bruto’ de color verdadero de 24 bits por píxel). El éxito de este procedimiento en este tipo de imágenes puede explicarse exactamente a partir de su carácter de paleta, que nos permite tener en cuenta variaciones locales agudas (¡incluso las no limítrofes!) de color, que surgen de la ‘mezcla’ de los puntos de las superficies frontales y traseras (que pueden diferir en gran medida, como en el caso del “Ángel”). El objetivo de buscar el escaneo óptimo es reducir estas variaciones tanto como sea posible.

3.3. Resultados de pruebas

Los ejemplos de modelos en los formatos originales y comprimidos se muestran en el Anexo 3. La calidad de algunos modelos (p. ej., el Ángel) no es todavía bastante satisfactoria después de la compresión, mientras que otras son muy buenas (‘Saltamontes’). Sin embargo, sentimos que este problema puede resolverse con ayuda de un escaneo adecuado. Potencialmente, incluso la modalidad de compresión por 12 podría usarse, por lo que la compresión global aumenta aún más. Finalmente, la compresión sin pérdidas mejorará a fin de aproximarse a los mejores resultados basados en PPM en la compresión geométrica.

Aquí damos una tabla de razones de compresión.

Modelo	Razón para el mejor procedimiento de PPM	Razón para el procedimiento sencillo de contexto-1
“Ángel”	7,1	6,7
“Morton”	7,5	6,7
“Saltamontes”	7,8	7,4

4. Conclusión

En este documento, se informa sobre el resultado del experimento central sobre Representación de Profundidad Basada en Imágenes, AFX A8.3. Se ha introducido el flujo de DIBR, que se enlaza a través de los campos de URL de los nodos de DIBR. Estos flujos consisten en todos los elementos en el nodo de DIBR, junto con un indicador para cada elemento, para hacer que sea optativo. Además, se investigan la compresión del octárbol y la de los datos de TexturaDePunto.

Anexo 1. Significado geométrico de la invariancia ortogonal del contexto en el algoritmo de compresión de BVO.

La hipótesis de invariancia ortogonal se ilustra en la FIG. 6. Consideremos la rotación alrededor del eje vertical en 90 grados en el sentido de las agujas del reloj. Consideremos los patrones arbitrarios de rellenado del nodo y su padre antes (imagen superior) y después de la rotación (imagen inferior). Luego, dos patrones distintos pueden tratarse como el mismo patrón.

Anexo 2. Grupos y Transformadas.

1. 32 transformadas ortogonales fijas.

Cada transformada se especifica por una palabra de 5 bits. La combinación de bits es la composición de las siguientes transformadas básicas (es decir, si el k-ésimo bit es 1, se realiza la correspondiente transformada)

1er bit – permutar coordenadas x e y;

2º bit – permutar coordenadas y y z;

3er bit – simetría en el plano (y-z);

4º bit – simetría en el plano (x-z);

5º bit – simetría en el plano (x-y);

2. 27 grupos.

Para cada grupo, aquí está el orden del grupo y el número de bits no nulos en sus elementos: NúmeroDeGrupo, CantidadDeGrupos y NúmeroDeBitsDeRelleno (ConjuntoVoxels).

Grupo	Orden del grupo (número de elementos)	Nº de bits no nulos en cada elemento del grupo
0	1	0
1	8	1
2	8	2
3	4	2
4	12	2
5	24	3
6	6	4
7	8	3
8	8	4

(cont.)		
9	4	2
10	24	3
11	16	4
12	8	4
13	24	4
14	24	5
15	4	4
16	16	5
17	8	6
18	2	4
19	8	5
20	4	6
21	2	4
22	8	5
23	12	6
24	4	6
25	8	7
26	1	8

3. Símbolos y transformadas.

Para cada símbolo (s), aquí está el índice del grupo (g) al que pertenece y el valor de la transformada (t) que lo lleva al elemento 'estándar' del grupo.

El número binario del símbolo se asocia a las coordenadas binarias de voxel de la siguiente manera: el i-ésimo bit del número tiene coordenadas binarias x=i&1, y=i&(1<<1), <=i&(1<<2).

s	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
g	0	1	1	2	1	3	4	5	1	4	3	5	2	5	5
t	0	0	4	0	8	0	0	0	12	4	4	4	8	8	12
s	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
g	6	1	2	4	5	4	5	7	8	9	10	10	11	10	12
t	0	16	2	1	1	2	2	0	0	0	0	5	0	10	0

...

s	241	242	243	244	245	256	247	248	249	250	251	252	253	254	255
g	14	14	17	14	20	23	25	14	23	20	25	17	25	25	26
t	16	20	16	24	16	16	16	28	20	20	20	24	24	28	0

Anexo 3. Pantallazos de compresión de TexturaDePunto.

En las FIGs. 7, 8 y 9, las cifras de compresión geométrica se dan para el mejor procedimiento basado en PPM.

III. Resultado del experimento central sobre Representación de Profundidad Basada en Imágenes (AFX A8.3)

1. Introducción

En este documento, se informa sobre el resultado del experimento central sobre Representación de Profundidad Basada en Imágenes (DIBR), AFX A8.3. Este experimento central es para los nodos de representación de profundidad basada en imágenes que usan texturas con información de profundidad. Los nodos han sido aceptados e incluidos en una propuesta para el Borrador del Comité durante la reunión de Pattaya. Sin embargo, el flujo de esta información por el nodo de ImagenOctárbol y el nodo de ImagenConProfundidad aún permanecían en proceso. Este documento describe el formato del flujo a enlazar por parte de estos nodos. El formato del flujo incluye la compresión del campo octárbol del nodo ImagenOctárbol y los campos de profundidad / color del nodo TexturaDePunto.

2. Compresión de formatos de DIBR

Describimos aquí una técnica novedosa para la compresión eficaz sin pérdida de una estructura de datos de octárbol sin enlace, que permite una reducción en el volumen de esta representación, ya compacta, de alrededor de entre 1,5 a 2 veces, en nuestros experimentos. También sugerimos varias técnicas para la compresión sin pérdida y con pérdida del formato de TexturaDePunto, usando una representación intermedia de voxel en combinación con la codificación por entropía y el procedimiento especializado de compresión de textura basada en bloques [6].

2.1. Compresión de ImagenOctárbol

Los campos de imágenesoctárbol y octárbol en la ImagenOctárbol se comprimen por separado. Los procedimientos descritos se han desarrollado en base a la noción de que el campo octárbol debe comprimirse sin pérdidas, mientras que se permite algún grado de distorsión visualmente aceptable para las imágenesoctárbol. El campo de imágenesoctárbol se comprime por medio de la compresión de imágenes del estándar MPEG-4 (para el modelo estático), o por herramientas de compresión de vídeo (para el modelo animado).

2.1.1. Compresión del campo octárbol

La compresión del octárbol es la parte más importante de la compresión de ImagenOctárbol, ya que trata de la compresión de una representación ya muy compacta de un árbol binario sin enlace. Sin embargo, en nuestros experimentos, el procedimiento explicado más adelante redujo el volumen de esta estructura a casi la mitad del original. En la versión animada de ImagenOctárbol, el campo Octárbol se comprime por separado para cada trama tridimensional.

2.1.1.1. Modelo de contexto

La compresión se realiza por una variante de la codificación aritmética adaptable (implementada como un ‘codificador de distancias’) que hace uso explícito de la naturaleza geométrica de los datos. El Octárbol es un flujo de octetos. Cada octeto representa a un nodo (es decir, subcubo) del árbol, en el cual sus bits indican la ocupación del subcubo después de la subdivisión interna. El patrón de bits se llama el patrón de relleno del nodo. El algoritmo de compresión descrito procesa los octetos uno por uno, de la siguiente manera.

* Se determina un contexto para el octeto actual.

* se recupera la ‘probabilidad’ (frecuencia normalizada) de ocurrencia del octeto actual en este contexto de la ‘tabla de probabilidades’ (PT) correspondiente al contexto.

* Se suministra el valor de probabilidad al codificador de distancias.

* La PT actual se actualiza añadiendo 1 a la frecuencia de la ocurrencia del octeto actual en el contexto actual (y, si es necesario, se renormaliza después; véanse los detalles más adelante).

Así, la codificación es el proceso de construir y actualizar las PT según el modelo de contexto. En los esquemas de codificación aritmética adaptable basados en el contexto (tales como la ‘Predicción con Coincidencia Parcial’), el contexto de un símbolo es usualmente una cadena de varios símbolos precedentes. Sin embargo, en nuestro caso, la eficacia de la compresión aumenta al explotar la estructura del octárbol y la naturaleza geométrica de los datos. El enfoque descrito se basa en dos ideas que son aparentemente nuevas en el problema de la compresión de un octárbol.

A. Para el nodo actual, el contexto es bien su nodo padre, o bien el par {nodo padre, posición actual del nodo en el nodo padre};

B. Se supone que la 'probabilidad' de la ocurrencia de un nodo dado en la ubicación geométrica específica en el nodo padre específico es invariante con respecto a un cierto conjunto de transformadas ortogonales (tales como rotaciones o simetrías).

La hipótesis 'B' se ilustra en la FIG. 6, para la transformada R, que es la rotación en -90º en el plano x-z. La noción básica detrás de 'B' es la observación de que la probabilidad de ocurrencia de un tipo específico de nodo hijo en un tipo específico de nodo padre debería depender solamente de su posición relativa. Esta hipótesis se confirma en nuestros experimentos, por el análisis de tablas de probabilidades. Nos permite usar un contexto más complejo, sin tener demasiadas tablas de probabilidades. Esto, a su vez, ayuda a lograr resultados bastante buenos en términos de tamaño y velocidad de los datos. Obsérvese que cuantos más contextos se usen, más precisa es la probabilidad estimada y, por tanto, más compacto es el código.

Introduzcamos el conjunto de transformadas para el cual supondremos la invariancia de distribuciones de probabilidad. A fin de aplicarse a nuestra situación, tales transformadas deberían preservar el cubo circundante. Consideremos un conjunto G de transformadas ortogonales en el espacio euclidiano, que se obtienen por todas las composiciones, en cualquier número y orden, de las 3 transformadas básicas (generadores) m_1, m_2 y m_3, dadas por

$$\mathbf{m}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{m}_3 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

donde $\mathbf{m}_1$ y $\mathbf{m}_2$ son reflejos sobre los planos x=y e y=z, respectivamente, y m_3 es el reflejo sobre el plano x=0. Uno de los resultados clásicos de la teoría de grupos generados por reflejos afirma que G contiene 48 transformadas ortogonales distintas, y es, en cierto sentido, el máximo grupo de transformadas ortogonales que convierten el cubo en sí mismo (el llamado grupo de Coxeter). Por ejemplo, la rotación R en la FIG. 1 se expresa, mediante los generadores, como

$$\mathbf{R} = \mathbf{m}_3 \cdot \mathbf{m}_2 \cdot \mathbf{m}_1 \cdot \mathbf{m}_2$$

donde '·' es la multiplicación matricial.

La transformada de G, aplicada a un nodo de octárbol, produce un nodo con un patrón distinto de rellenado de subcubos. Esto nos permite categorizar los nodos según el patrón de rellenado de sus subcubos. Usando el lenguaje de la teoría de grupos, decimos que G actúa sobre el conjunto de todos los patrones de rellenado de los nodos del octárbol. Los cálculos muestran que existen 22 clases distintas (también llamadas órbitas en la teoría de grupos), en las cuales, por definición, dos nodos pertenecen a la misma clase si, y sólo si, están conectados por una transformada de G. El número de elementos en una clase varía entre 1 y 24, y siempre es un divisor de 48.

La consecuencia práctica de 'B' es que la tabla de probabilidades depende no del nodo padre en sí, sino solamente de la clase a la cual pertenece el nodo padre. Obsérvese que habría 256 tablas para un contexto basado en padres y unas 256x8 = 2048 tablas adicionales para un contexto basado en la posición de padre e hijo en el caso anterior, mientras que solamente necesitamos 22 tablas para el contexto basado en la clase del padre, más 22x8 = 176 tablas en el último caso. Por lo tanto, es posible usar un contexto equivalentemente complejo con un número relativamente pequeño de tablas de probabilidades. La PT construida tendría la forma que se muestra en la Tabla 2.

Tabla 2. Enumeración de tablas de probabilidades.

Identificador de las PT	0	1	...	255	Descripción del contexto
0	P0,0	P0,1	...	P0,255	Contexto-0: Independiente del contexto
1..22(22)	Pi,0	Pi,1	...	Pi,255	Contexto-1: {clase del nodo padre}
23..198(176)	Pj,0	Pj,1	...	Pj,255	Contexto-2: {clase del nodo padre, posición actual del nodo}

2.1.1.2. Proceso de codificación

Para hacer que las estadísticas para las tablas de probabilidad sean más precisas, se recogen de distintas maneras en tres etapas del proceso de codificación.

* En la primera etapa no usamos contextos en absoluto, aceptando el 'modelo de contexto-0', y mantenemos una única tabla de probabilidades con 256 entradas, a partir de la distribución uniforme;

* En cuanto se codifican los primeros 512 nodos (es un número hallado empíricamente), conmutamos al 'modelo de contexto-1', usando el nodo padre como un contexto. En el momento de la conmutación, la PT del contexto-0 se copia a las PT para todos los 22 contextos.

* Después de que se han codificado 2.048 nodos (otro valor heurístico), conmutamos al 'modelo de contexto-2'. En este momento, las PT de contexto-1 de los patrones padres se copian a las PT para cada posición en el mismo patrón padre.

El punto clave del algoritmo es la determinación del contexto y la probabilidad para el octeto actual. Esto se implementa de la siguiente manera. En cada clase fijamos un elemento individual, que se llama 'elemento estándar'. Almacenamos una tabla de mapas de clases (CMT) que indica la clase a la cual pertenece cada uno de los posibles 256 nodos, y la transformada precalculada de G que lleva a este nodo específico al elemento estándar de su clase. Así, a fin de determinar la probabilidad del nodo actual N, llevamos a cabo las siguientes etapas:

* Mirar al padre P del nodo actual;

* Recuperar la clase de la CMT, a la cual pertenece P, y la transformada T que lleva a P al nodo estándar de la clase.

* Permitir que sea c el número de la clase;

* Aplicar T a P y hallar la posición p del hijo en el nodo estándar al cual se asocia el nodo actual N;

* Aplicar T a N. Luego, el patrón TN de relleno recién obtenido está en la posición p en el nodo estándar de la clase c.

* Recuperar la probabilidad requerida de la entrada TN de la tabla de probabilidades correspondiente a la combinación (c, p) de clase y posición.

Para el modelo del contexto-1, las etapas anteriores se modifican de manera obvia. No hace falta decir que todas las transformadas están precalculadas e implementadas en una tabla de búsqueda.

Obsérvese que en la etapa de descodificación del nodo N, su padre P ya está descodificado y, por tanto, la transformada T se conoce. Todas las etapas en la etapa de descodificación son absolutamente similares a las correspondientes etapas de codificación.

Finalmente, esbocemos el proceso de actualización de probabilidades. Sea P una tabla de probabilidades para algún contexto. Indiquemos con P(N) la entrada de P correspondiente a la probabilidad de ocurrencia del nodo N en este contexto. En nuestra implementación, P(N) es un entero y, después de cada ocurrencia de N, P(N) se actualiza como:

$$P(N)=P(N)+A,$$

donde A es un parámetro incremental entero que varía habitualmente entre 1 a 4 para distintos modelos de contexto. Sea S(P) la suma de todas las entradas en P. Entonces, la 'probabilidad' de N que se suministra al codificador aritmético (codificador de distancias en nuestro caso) se calcula como P(N) / S(P). En cuanto S(P) alcanza un valor de umbral 2^{16}, todas las entradas se renormalizan: a fin de evitar la ocurrencia de valores cero en P, las entradas iguales a 1 se dejan intactas, mientras que las otras se dividen entre 2.

2.2. Compresión de TexturaDePunto

El nodo TexturaDePunto contiene dos campos a comprimir, esto es, profundidad y color. Las principales dificultades en la compresión de datos de TexturaDePunto se deben a los siguientes requisitos:

* La geometría debe comprimirse de manera que no tenga pérdidas, ya que las distorsiones en este tipo de representación geométrica son a menudo sumamente notables.

* La información de color no tiene ninguna estructura bidimensional natural y, por ello, las técnicas de compresión de imágenes no son inmediatamente aplicables.

En esta sección sugerimos tres procedimientos para la compresión del modelo de TexturaDePunto:

* Procedimiento sin pérdidas para la representación estándar del nodo.

* Procedimiento sin pérdidas para la representación de resolución inferior del nodo.

* Geometría sin pérdidas y compresión de color con pérdidas para la representación de resolución inferior del nodo.

Los procedimientos corresponden a los tres niveles de 'fidelidad' de la descripción del objeto. El primer procedimiento supone que debemos almacenar la información de profundidad hasta su precisión original de 32 bits. Sin embargo, en la práctica, la información de profundidad puede cuantizarse a menudo en un número mucho más pequeño de bits, sin pérdida de calidad. En particular, cuando el modelo de TexturaDePunto se convierte a partir del modelo poligonal, se escoge la resolución de la cuantización según el tamaño efectivo de los detalles visibles que posee el modelo original, así como la resolución deseable de la pantalla de salida. En este caso, entre 8 y 11 bits bien pueden satisfacer los requisitos, y los valores de profundidad se almacenan inicialmente en este formato de resolución inferior. Ahora, nuestro segundo procedimiento trata la compresión sin pérdidas de esta representación de 'resolución inferior'. La observación clave aquí es que, para un tal número relativamente pequeño (en comparación a los 32 estándar) de bits, puede usarse una representación intermedia de voxels del modelo, que nos permite comprimir el campo de profundidad, esencialmente sin pérdida de información. La información de color en ambos casos se comprime sin pérdidas y se almacena en un formato de PNG, después de redisponer los datos de color como una imagen bidimensional auxiliar.

Finalmente, el tercer procedimiento nos permite lograr una compresión mucho mayor, combinando la compresión sin pérdidas de la geometría con la compresión con pérdidas de los datos de color. Esto último es realizado por una técnica especializada de compresión de texturas basada en bloques. En las siguientes tres subsecciones, se describen los procedimientos en todo detalle.

2.1.1. Compresión sin pérdidas de TexturaDePunto para la representación estándar del nodo

Este es un procedimiento sencillo de codificación sin pérdida, que funciona de la siguiente manera.

* el campo de profundidad es comprimido por el codificador de distancias adaptable, similar al usado en la compresión de campos del Octárbol. Para este formato, usamos una versión en la cual la tabla de probabilidades se mantiene para cada uno de los contextos de 1 símbolo, y el contexto es sencillamente el octeto anterior. Por lo tanto, se usan 256 PT. El campo de profundidad se considera como un flujo de octetos, y la estructura geométrica no se usa explícitamente.

* El campo color se comprime después de la conversión en una imagen plana de color auténtico. Los colores de los puntos en el modelo de TexturaDePunto se graban primero en una formación unidimensional temporal, en el mismo orden que los valores de profundidad en el campo profundidad. Si el número total de puntos en el modelo es L, entonces calculamos el entero I más pequeño tal que $I \cdot I \geq L$, y 'plegamos' esta larga 'cadena' de valores de color en la imagen cuadrada de lado I (si es necesario, rellenando con píxeles negros). Esta imagen se comprime luego con una de las herramientas de compresión de imágenes sin pérdidas del estándar MPEG-4. En nuestro enfoque, usamos un formato de Gráficos Portátiles en Red (PNG). La imagen obtenida de esta manera a partir del modelo del 'Ángel' se muestra en la FIG. 10 (a).

2.2.2. Compresión sin pérdidas de la TexturaDePunto para la representación de resolución inferior del nodo

En muchos casos, la resolución de 16 bits para la información de profundidad es excesivamente fina. De hecho, la resolución en profundidad debería corresponder a la resolución de la pantalla sobre la cual el modelo ha de visualizarse. En situaciones donde pequeñas variaciones en la profundidad del modelo en distintos puntos lleven a un desplazamiento en el plano de la pantalla mucho más pequeño que el tamaño del píxel, es razonable usar una resolución inferior en profundidad, y los modelos a menudo se representan en el formato en que los valores de profundidad ocupan entre 8 y 11 bits. Tales modelos se obtienen usualmente a partir de otros formatos, p. ej., el modelo poligonal, discretizando los valores de profundidad y color en la rejilla espacial adecuada.

Tal representación de resolución reducida puede considerarse en sí misma como una forma comprimida del modelo estándar con profundidad de 32 bits. Sin embargo, existe una representación más compacta para tales modelos, usando el espacio intermedio de voxels. En efecto, puede suponerse que los puntos del modelo pertenecen a nodos de rejilla espacial uniforme, con la separación determinada por el paso de discretización. Siempre podemos suponer que la rejilla es uniforme y ortogonal, ya que, en caso del modelo en perspectiva, podemos trabajar en el espacio paramétrico. Usando esta observación, los campos de profundidad y color de la TexturaDePunto de resolución inferior se comprimen de la siguiente manera.

* el campo color se comprime por medio de una técnica de compresión de imágenes sin pérdida, como en el procedimiento anterior;

* el campo profundidad se transforma primero a la representación por voxels, y luego se comprime por la variante del codificador de distancias descrito en la subsección anterior.

El modelo intermedio de voxels se construye de la siguiente manera. Según la resolución s de profundidad del modelo, consideremos el espacio discreto de voxels de tamaño *ancho x altura x* 2^s (los parámetros 'ancho' y 'altura' se explican en la especificación de la TexturaDePunto). Para nuestros fines, no necesitamos trabajar con un espacio de voxels potencialmente enorme como un todo, sino solamente con sus secciones transversales 'delgadas'. Indiquemos con (r, c)

las coordenadas de fila y columna del modelo en el plano de proyección, y sea d la coordenada de profundidad. Transformamos las 'tajadas' {c = constante}, es decir, las secciones transversales del modelo por 'planos verticales', en la representación por voxels. Recorriendo la tajada a lo largo de las 'columnas' paralelas al plano de proyección, fijamos el voxel (r, c, d) en 'negro' si y sólo si existe un punto del modelo con un valor d de profundidad que se proyecte en (r, c). El proceso se ilustra en la FIG. 4.

En cuanto la tajada está construida, es comprimida por el codificador de distancias de contexto-1, y comienza la compresión de la siguiente tajada. De esta manera, evitamos trabajar con formaciones muy grandes. Las tablas de probabilidades no se inicializan para cada nueva tajada. Para una amplia gama de modelos solamente una pequeña fracción de los voxels son negros, y esto nos permite lograr una razón de compresión bastante alta. La descompresión se efectúa por la inversión obvia de las operaciones descritas.

Se describirá la comparación entre la compresión del campo profundidad por este procedimiento y por la representación de octárbol. La razón global de compresión del modelo está determinada, sin embargo, por el campo color, ya que una tal imagen irregular no puede comprimirse extremadamente sin distorsiones. En la próxima subsección consideramos una combinación de geometría sin pérdida y de técnica de compresión de color con pérdida.

2.2.3. Geometría sin pérdida y compresión de color con pérdida para la representación de TexturaDePunto de resolución inferior

Como el anterior, este procedimiento transforma el campo profundidad en la representación por voxels, que es comprimida luego por el codificador adaptable de distancias de contexto-1. El campo color también se asocia a la imagen bidimensional. Sin embargo, hacemos un intento de organizar la asociación de modo tal que los puntos que estén cercanos en espacio tridimensional se asocien a puntos cercanos en el plano de imágenes bidimensionales. Luego se aplica un procedimiento especializado de compresión de textura (particiones adaptables de bloques, ABP) a la imagen resultante. Las etapas principales del algoritmo son las siguientes.

1. Transformar una 'tajada' de cuatro 'planos verticales' sucesivos del modelo de TexturaDePunto en la representación por voxels.

2. Escanear la formación obtenida de voxels de dimensiones *ancho x 4 x* 2^5:

* Recorriendo el 'plano' vertical de subcubos de voxeles de 4 x 4 x 4 a lo largo de las 'columnas' paralelas al plano de proyección: primero la columna más cercana al plano de proyección, luego la siguiente columna más cercana, etc. (es decir, en el orden usual de recorrido de formaciones bidimensionales).

* Recorriendo los voxels dentro de cada subcubo de 4x4x4 en el orden análogo al usado en el recorrido de los subcubos de nodos de ImagenOctárbol.

3. Grabar los colores de los puntos del modelo encontrados en este orden de recorrido, en una formación unidimensional auxiliar;

4. Redisponer la formación obtenida de colores en una imagen bidimensional, de modo que:

5. 64 muestras consecutivas de color se dispongan, por columnas, en un bloque de píxeles de 8 por 8, 64 muestras siguientes se dispongan en el bloque adyacente de píxeles de 8 por 8, y así sucesivamente.

6. Comprimir la imagen obtenida por la técnica ABP.

Este procedimiento de escaneo de una formación tridimensional y de transformación del resultado en la imagen bidimensional se escogió a partir de las siguientes consideraciones. Obsérvese que los subcubos de 4x4x4 y los bloques de imagen 8x8 contienen el mismo número de muestras. Si varios subcubos sucesivamente escaneados contienen suficientes muestras de color como para llenar el bloque 8x8, es sumamente probable que este bloque sea bastante uniforme y, por tanto, la distorsión será apenas discernible en el modelo tridimensional después de la descompresión. El algoritmo ABP comprime bloques 8x8 independientemente de otros, con la ayuda de la paletización local. En nuestras pruebas, la distorsión introducida por la compresión ABP en el modelo tridimensional final fue drásticamente menor que la del estándar JPEG. Otra razón para escoger este algoritmo fue la gran velocidad de descompresión (para la cual fue diseñado originalmente). La razón de compresión puede tomar dos valores, 8 y 12. En el algoritmo de compresión de TexturaDePunto, fijamos la razón de compresión en 8.

Desafortunadamente, este algoritmo no es universalmente aplicable. Aunque la imagen obtenida de esta manera a partir del campo color, mostrada en la FIG. 10 (b), es mucho más uniforme que para el orden 'natural' de escaneo, a veces los bloques bidimensionales de 8x8 pueden contener muestras de color correspondientes a puntos distantes en el espacio tridimensional. En este caso, el procedimiento ABP con pérdida puede 'mezclar' colores de distintas partes del modelo, lo que lleva a una distorsión local pero notable después de la descompresión.

Sin embargo, para muchos modelos, el algoritmo funciona muy bien. En la FIG. 11, mostramos el caso 'malo' (modelo 'Ángel') y el caso 'bueno' (modelo 'Morton256'). La reducción del volumen del modelo en ambos casos es de alrededor de 7 veces.

3. Resultados de pruebas

En esta sección comparamos los resultados de la compresión de dos modelos, 'Ángel' y 'Morton256', en dos formatos distintos: ImagenOctárbol y TexturaDePunto. Las dimensiones de las imágenes de referencia para cada modelo fueron de 256x256 píxeles.

3.1. Compresión de TexturaDePunto

En las Tablas 3 a 5, se dan los resultados de distintos procedimientos de compresión. Los modelos para este experimento se obtuvieron de modelos con un campo de profundidad de 8 bits. Los valores de profundidad se expandieron sobre la gama $(1, 2^{30})$, usando el paso de cuantización $2^{21}+1$, a fin de hacer la distribución de los bits en los valores de profundidad de 32 bits más uniforme, imitando en cierto grado los valores 'verdaderos' de 32 bits.

No han de esperarse grandes razones de compresión de este procedimiento. La reducción del volumen es del mismo orden que para la típica compresión sin pérdida de las imágenes de color verdadero. Los campos comprimidos de profundidad y color son de tamaño bastante comparable, ya que la naturaleza geométrica de los datos no es capturada por este enfoque.

Veamos ahora cuánto pueden comprimirse sin pérdida los mismos modelos cuando se toman con su 'verdadera' resolución de profundidad. A diferencia del caso anterior, el campo profundidad se comprime sin pérdida alrededor de entre 5 y 6 veces. Esto se debe a la representación intermedia por voxels que hace la redundancia geométrica de los datos mucho más pronunciada: en efecto, sólo una pequeña fracción de los voxels son negros. Sin embargo, dado que el tamaño no comprimido de los modelos es más pequeño que para el caso de 32 bits, la razón de compresión del campo color determina ahora la razón global de compresión, que es incluso más pequeña que para el caso de 32 bits (aunque los ficheros de salida son también más pequeños). Luego, es deseable poder comprimir el campo color al menos tan bien como el campo profundidad.

Nuestro tercer procedimiento usa la técnica de compresión con pérdida llamada ABP [6] para este fin. Este procedimiento da una compresión mucho mayor. Sin embargo, como todas las técnicas de compresión con pérdida, puede llevar a distorsiones desagradables en algunos casos. Un ejemplo de un objeto para el cual ocurre esto es el modelo 'Ángel'. En el proceso de escaneo de los puntos del modelo, los puntos espacialmente distantes caen a veces en el mismo bloque de imagen bidimensional. Los colores en puntos distantes de este modelo pueden diferir mucho, y la paletización local no puede proporcionar una aproximación exacta si hay demasiados colores distintos en un bloque. Por otra parte, es la paletización local lo que nos permite comprimir con precisión una vasta mayoría de los bloques, para los cuales la distorsión introducida por, digamos, el estándar JPEG se torna absolutamente insoportable después de que los colores reconstruidos se recolocan en sus ubicaciones tridimensionales. Sin embargo, la calidad visual del modelo 'Morton256' comprimido por el mismo procedimiento es excelente, y este fue el caso para la mayoría de los modelos en nuestros experimentos.

Tabla 3. Compresión sin pérdida de TexturaDePunto para el campo profundidad de 32 bits (En Octetos).

Modelo		campo profundidad	campo color	Tamaño total	Razón de compresión		
					Profundidad	Color	Total
"Morton256"	Original	691.032	321.666	1.012.698	3,1	1,2	2,0
	Comprimido	226.385	270.597	424.562			
"Ángel"	Original	665.488	302.508	967.996	3,3	1,2	2,1
	Comprimido	204.364	262.209	466.604			

Tabla 4. Compresión sin pérdida de TexturaDePunto para la representación de nodo de resolución inferior (En Octetos).

Modelo		campo profundidad	campo color	Tamaño total	Razón de compresión		
					Profundidad	Color	Total
"Morton256"	Original	172.758	321.666	494.424	5,4	1,2	1,63
	Comprimido	31.979	270.597	302.576			
"Ángel"	Original	166.372	302.508	468.880	5,2	1,2	1,6
	Comprimido	32.047	262.209	294.256			

Tabla 5. Geometría sin pérdida y compresión de color con pérdida para la TexturaDePunto de resolución inferior (En Octetos).

Modelo		campo profundidad	campo color	Tamaño total	Razón de compresión		
					Profundidad	Color	Total
"Morton256"	Original	172.758	321.666	494.424	5,4	8,0	6,8
	Comprimido	31.979	40.352	72.331			
"Ángel"	Original	166.372	302.508	468.880	5,2	7,9	6,7
	Comprimido	32.047	38.408	70.455			

3.2. Compresión de ImagenOctárbol

La Tabla 6 presenta tamaños de componentes comprimidos y no comprimidos de octárbol para nuestros dos modelos de prueba. Vemos que la reducción de este campo es de alrededor de entre 1,6 y 1,9 veces.

Sin embargo, en comparación con los modelos no comprimidos de TexturaDePunto, incluso con el campo profundidad de 8 bits, ImagenOctárbol es mucho más compacta. La Tabla 7 muestra las razones de compresión 7,2 y 11,2. Esto es más de lo que las TexturasDePunto pueden comprimirse sin convertirse en ImagenOctárbol (6,7 y 6,8 veces, respectivamente). Sin embargo, como ya hemos mencionado, ImagenOctárbol puede contener información de color incompleta, que es el caso en el modelo 'Ángel'. En tales casos se usa la interpolación tridimensional de colores.

Para resumir, podemos concluir que los experimentos presentados anteriormente demuestran la eficacia de las herramientas de compresión desarrolladas. La selección de la mejor herramienta para un modelo dado depende de su complejidad geométrica, del carácter de la distribución del color, de la velocidad requerida de representación y de otros factores.

Tabla 6. Razones de compresión obtenidas por el procedimiento descrito en 4.1.2, para modelos de ImagenOctárbol y sus componentes (tamaños de fichero redondeados a Koctetos).

Modelo	Tamaño de Octárbol	Tamaño comprimido de Octárbol	Razón de compresión
"Ángel"	50	31	1.6
"Morton256"	41	22	1,9

Tabla 7. TexturaDePunto no comprimida (campo profundidad de 8 bits) y representaciones comprimidas de ImagenOctárbol para los mismos modelos (tamaños de fichero redondeados a Koctetos).

Modelo	TexturaDePunto	ImagenOctárbol comprimida	Razón de compresión
“Ángel”	469	65	7.2
“Morton256”	494	44	11,2

5. Observaciones sobre el estudio de ISO/IEC 14496-1/PDAM4

Después de aplicar las siguientes revisiones al Estudio de ISO/IEC 14496-1/PDAM4 (N4627), el Estudio revisado de ISO/IEC 14496-1/PDAM4 debería incorporarse a ISO/IEC 14496-1/FPDAM4.

Cláusula 6.5.3.1.1, Técnica

Problema: El valor por omisión del campo ortográfico debería ser el valor usado más generalmente.

Solución: reemplazar el valor por omisión del campo ortográfico “FALSO” por “VERDADERO”, según lo siguiente.

Revisión propuesta:

campo SFBool ortográfico VERDADERO

Cláusula 6.5.3.1.1, Técnica

Problema: El flujo de DIBR se hará con el procedimiento de flujo uniforme para AFX.

Solución: Eliminar el campo URLImagenConProfundidad del nodo ImagenConProfundidad.

Revisión propuesta:

ImagenConProfundidad {

campo	SFVec3f	posición	0 0 10
campo	SFRotation	orientación	0 0 1 0
campo	SFVec2f	campoDeVisión	0,785398 0,785398
campo	SFFloat	PlanoCercano	10
campo	SFFloat	PlanoLejano	100
campo	SFBool	ortográfico	VERDADERO
campo	SFNode	diTextura	NULO

}

Cláusula 6.5.3.1.2, De edición

Problema: El término ‘normalizado’ confunde, según se aplica al campo profundidad en el contexto actual.

Solución: En el 5º párrafo, cambiar ‘normalizado’ por ‘ajustado’.

Revisión propuesta:

Los campos PlanoCercano y PlanoLejano especifican las distancias desde el punto de vista al plano cercano y al plano lejano del área de visibilidad. Los datos de textura y profundidad muestran el área encerrada por el plano cercano, el plano lejano y el campoDeVisión. Los datos de profundidad se ajustan a la distancia entre el PlanoCercano y el PlanoLejano.

Cláusula 6.5.3.1.2, Técnica

Problema: El flujo de DIBR se hará con el procedimiento de flujo uniforme para AFX.

Solución: Eliminar la explicación del campo URLImagenConProfundidad (el 7º párrafo y siguientes).

Revisión propuesta:

Cláusula 6.5.3.2.2, De edición

Problema: La semántica del campo profundidad está especificada de forma incompleta.

Solución: Cambiar la especificación del campo profundidad en el 3er párrafo, según lo siguiente.

Revisión propuesta:

El campo profundidad especifica la profundidad para cada píxel en el campo textura. El tamaño del mapa de profundidades será el mismo tamaño que el de la imagen o película en el campo textura. El campo profundidad será uno de los diversos tipos de nodos de textura (TexturaDeImagen, TexturaDePelícula o TexturaDePíxel), donde sólo se permiten los nodos que representan imágenes en escala de grises. Si el campo profundidad no está especificado, se usará el canal alfa en el campo textura como el mapa de profundidades. Si el mapa de profundidades no está especificado a través del campo profundidad o el canal alfa, el resultado está indefinido.

El campo profundidad nos permite calcular la distancia efectiva de los puntos tridimensionales del modelo al plano que atraviesa el punto de vista, y es paralelo al plano cercano y al plano lejano:

$$dist = PlanoCercano + \left(1 - \frac{d-1}{d_{max}-1}\right)(PlanoLejano - PlanoCercano),$$

donde d es el valor de profundidad y d_{max} es el máximo valor admisible de profundidad. Se supone que, para los puntos del modelo, d > 0, donde d = 1 corresponde al plano lejano y d = d_{max} corresponde al plano cercano.

Esta fórmula es válida tanto para el caso de perspectiva como para el ortográfico, ya que d es la distancia entre el punto y el plano. d_{max} es el mayor valor de d que puede representarse con los bits usados para cada píxel:

(1) Si la profundidad se especifica mediante el campo profundidad, entonces el valor d de profundidad es igual a la escala de grises.

(2) Si la profundidad se especifica mediante el canal alfa en la imagen definida mediante el campo textura, entonces el valor de profundidad es igual al valor del canal alfa.

El valor de profundidad también se usa para indicar qué puntos pertenecen al modelo: sólo el punto para el cual d es no nulo pertenece al modelo.

Para el modelo animado basado en ImagenConProfundidad, sólo se usa ImagenConProfundidad con TexturasSimples como diTexturas.

Cada una de las Texturas Simples puede animarse de una de las siguientes maneras:

(1) el campo profundidad es una imagen fija que satisface la condición anterior, y el campo textura es una TexturaDePelícula arbitraria

(2) el campo profundidad es una TexturaDePelícula arbitraria que satisface la condición anterior sobre el campo profundidad, y el campo textura es una imagen fija

(3) tanto profundidad como textura son TexturasDePelícula, y el campo profundidad satisface la condición anterior

(4) el campo profundidad no se usa, y la información de profundidad se recupera del canal alfa de la TexturaDePelícula que anima el campo textura

Cláusula 6.5.3.3.2, De edición

Problema: La semántica del campo profundidad está especificada de forma incompleta.

Solución: Reemplazar la especificación del campo profundidad (3er párrafo) por la revisión propuesta.

Revisión propuesta:

El significado geométrico de los valores de profundidad, y todas las convenciones sobre su interpretación, adoptadas para la TexturaSimple, valen aquí también.

El campo profundidad especifica profundidades múltiples de cada punto en el plano de proyección, que se supone que es el PlanoLejano (véase anteriormente) en el orden de recorrido, que comienza en el punto en la esquina inferior izquierda y

avanza hacia la derecha para acabar la línea horizontal, antes de avanzar a la línea superior. Para cada punto, el número de profundidades (píxeles) se almacena primero, y vendrán a continuación ese número de valores de profundidad.

Cláusula 6.5.3.4.1, H.1, Técnica

Problema: El tipo de campo SFString, usado para el campo octárbol, podría conducir a valores incoherentes

Solución: Cambiar el tipo de campo para el campo octárbol por MFInt32

Revisión propuesta:

En la cláusula 6.5.3.4.1

campo MFInt32 octárbol

En la cláusula H.1, tabla para Octárbol, cambiar la columna del octárbol según lo siguiente:

Nombre del campo		Identificador de DEF	Identificador de Entrada	Identificador de SALIDA	Identificador DYN	[m,M]	Q	A
octárbol	I MFInt32	01				[0,255]	13,8	

Cláusula 6.5.3.4.1, Técnica

Problema: El flujo de DIBR se hará con el procedimiento de flujo uniforme para AFX.

Solución: Eliminar el campo URLOctárbol del nodo ImagenOctárbol.

Revisión propuesta:

ImagenOctárbol {

campo SFInt32 resoluciónoctárbol 256

campo MFInt32 octárbol “”

campo MFNode imágenesoctárbol []

}

Cláusula 6.5.3.4.2, De edición

Problema: la definición del campo resoluciónoctárbol (2º párrafo) admite errores de interpretación.

Solución: Revisar la descripción añadiendo la palabra ‘admisible’

Revisión propuesta:

El campo resoluciónoctárbol especifica el máximo número admisible de hojas del octárbol a lo largo de un lado del cubo circundante.

El nivel del octárbol puede determinarse a partir de resoluciónoctárbol, usando la siguiente ecuación: niveloctárbol = int(log2(resoluciónoctárbol-1))+1)

Cláusula 6.5.3.4.2, Técnica

Problema: El flujo de DIBR se hará con el procedimiento de flujo uniforme para AFX,

Solución: Eliminar la explicación del campo URLOctárbol (el 5º párrafo y siguientes).

Revisión propuesta:

Cláusula 6.5.3.4.2, De edición

Problema: La animación de la ImagenOctárbol se describió de forma incompleta.

Solución: Añadir un párrafo al final de la cláusula 6.5.3.4.2 que describa la animación de ImagenOctárbol

Revisión propuesta:

La animación de la ImagenOctárbol puede llevarse a cabo mediante el mismo enfoque que en las tres primeras formas de la animación basada en ImagenConProfundidad, descrita anteriormente, con la única diferencia del uso del campo octárbol en lugar del campo profundidad.

Cláusula H.1, Técnica

Problema: La gama de datos de profundidad en el nodo de TexturaDePunto puede ser demasiado pequeña para aplicaciones futuras. Muchas herramientas gráficas admiten una profundidad de 24 bits o de 36 bits para su almacén temporal z. Sin embargo, el campo profundidad en la TexturaDePunto tiene la gama de [0, 65.535], que es de 16 bits.

Solución: En la cláusula H.1, tabla para TexturaDePunto, cambiar la gama de la columna de profundidad según lo propuesto.

Revisión propuesta:

Nombre del campo		Identificador de DEF	Identificador de Entrada	Identificador de SALIDA	Identificador DYN	[m,M]	Q	A
Profundidad	I MFInt32	10				[0,I]		

IV. ISO/IEC JTC 1/SC 29/WG 11 – CODIFICACIÓN DE PELÍCULAS Y AUDIO

1. Introducción

En este documento, se describe una mejora de ImagenOctárbol en la Representación Basada en Imágenes con Profundidad (DIBR), AFX A8.3. El nodo ImagenOctárbol ha sido aceptado e incluido en una propuesta para Borrador de Comité durante la reunión de Pattaya, PDAM de ISO / IEC 14496-1 / AMD4, " Org. Internacional para la Estandarización, N4415, diciembre de 2001, páginas 1-287 (XP001089813). Sin embargo, se ha observado que la calidad de representación sería insatisfactoria en algunos casos especiales, debido a la oclusión de la geometría del objeto. Este documento describe la versión mejorada del nodo ImagenOctárbol, es decir, el Octárbol Volumétrico Binario con Textura (TBVO), así como su procedimiento de compresión para el envío de flujos.

2. Octárbol Volumétrico Binario con Textura (TBVO)

2.1. Panorama del TBVO

El objetivo del TBVO es idear un formato más flexible de representación / compresión, con visualización rápida, como una mejora del Octárbol Volumétrico Binario (BVO). Esto se logra almacenando alguna información adicional en base al BVO. La representación basada en BVO consiste en (estructura de octárbol + conjunto de imágenes de referencia), mientras que la representación basada en TBVO consiste en (estructura de octárbol BVO + conjunto de imágenes de referencia + índices de cámara).

El principal problema de visualización del BVO es que debemos determinar el correspondiente índice de cámara de cada voxel durante la representación. Con este fin, necesitamos no solamente proyectar a las cámaras, sino también llevar a cabo el procedimiento inverso de difusión de rayos. Al menos, debemos determinar la existencia de una cámara, desde la cual sea visible el voxel. Por lo tanto, debemos hallar todos los voxels que se proyectan hacia una cámara específica. Pero este procedimiento es muy lento si usamos el enfoque de fuerza bruta. Hemos desarrollado un algoritmo que lo realiza rápida y exactamente para la mayoría de las formas de objetos. Sin embargo, aún hay algunos problemas para los voxels que no son visibles desde ninguna cámara.

Una posible solución podría ser almacenar el color explícito para cada voxel. Sin embargo, en este caso, hemos experimentado algún problema al comprimir información de color. Es decir, si agrupamos colores de voxel como un formato de imagen y lo comprimimos, la correlación de colores de los voxels vecinos se destruye, de modo tal que la razón de compresión sea insatisfactoria.

En el TBVO, el problema se resuelve almacenando el índice de (imagen de) cámara para cada voxel. El índice es usualmente el mismo para grandes grupos de voxels, y esto permite el uso de una estructura de octárbol para el almacenamiento económico de la información adicional. Obsérvese que, en promedio, solamente se observó un aumento de volumen del 15% en los experimentos con nuestros modelos. Su modelización es un poco más compleja, pero admite una manera más flexible de representar objetos de cualquier geometría.

Las ventajas del TBVO sobre el BVO son que su representación es más sencilla y mucho más rápida que la de los BVO, y

que no se impone virtualmente ninguna restricción sobre la geometría del objeto.

2.2. Ejemplo de TBVO

En esta sección mostramos un ejemplo típico, que ilustra la eficacia y los ingredientes claves de la representación del TBVO. En la FIG. 12 (a), se muestra un modelo de BVO del “Ángel”. Usando las usuales 6 texturas del BVO, unas pocas partes del cuerpo y del ala no son observadas desde ninguna cámara, produciendo una imagen representada con un montón de ‘grietas’ visibles. En la representación del TBVO del mismo modelo, se usa un total de 8 cámaras (6 caras de una caja + 2 cámaras adicionales). En la FIG. 13, (a) es la imagen del índice de cámara. El color distinto indica el índice distinto de la cámara. Cámaras adicionales están situadas dentro del cubo, observando la cara frontal y trasera ortográficamente. En la FIG. 13, (b) y (c) son imágenes adicionales tomadas por las cámaras adicionales. Como resultado, hemos obtenido un resultado de representación claro y sin fisuras del modelo, según se muestra en la FIG. 12 (b).

2.3. Descripción del flujo no comprimido del TBVO

Suponemos que 255 cámaras son suficientes, y asignamos hasta 1 octeto para el índice. El flujo de TBVO es un flujo de símbolos. Cada símbolo de TBVO es un símbolo de BVO o un símbolo de Textura. El símbolo de Textura indica un índice de cámara, que podría ser un número específico o un código de “indefinido”. Sea ‘?’ el código “indefinido” para la descripción posterior.

El flujo de TBVO se recorre en orden de amplitud primero. Describamos cómo grabar el flujo de TBVO si tenemos un BVO y cada voxel de nodo terminal tiene número de cámara. Esto debe hacerse en la etapa de modelización. Recorrerá todos los nodos del BVO, incluyen los nodos terminales (que no tienen símbolo de BVO) en orden de amplitud primero. El siguiente seudocódigo completará la grabación del flujo.

Si NodoAct no es nodo terminal

{ Grabar símbolo-BVO actual correspondiente a este nodo

si todos los hijos tienen idéntico índice de cámara (símbolo-de-textura)

Si padre de NodoAct tiene índice de cámara ‘?’

Grabar igual índice de cámara para subnodos}

en caso contrario

{ Grabar símbolo ‘?’ }

Según el procedimiento, para el árbol de TBVO mostrado en la FIG. 14 (a), puede obtenerse un flujo de símbolos según se muestra en la FIG. 14 (b). En este ejemplo, los símbolos de textura se representan en octetos. Sin embargo, en el flujo efectivo, cada símbolo de textura solamente necesitaría 2 bits, porque solamente necesitamos representar tres valores (dos cámaras y el código indefinido).

2.4. Compresión de TBVO

Los campos de imágenesoctárbol y de octárbol, en el nodo ImagenOctárbol, se comprimen por separado. Los procedimientos descritos han sido desarrollados en base al concepto de que el campo octárbol debe comprimirse sin pérdida, mientras que se admite algún grado de distorsión visualmente aceptable para las imágenesoctárbol.

2.4.1. Compresión del campo imágenesoctárbol

El campo imágenesoctárbol se comprime por medio de la compresión de imágenes del MPEG-4 (para el modelo estático), o de herramientas de compresión de vídeo (para el modelo animado) que se permitan en el estándar MPEG-4. En nuestro enfoque, usamos el formato JPEG para las ImágenesOctárbol (después de algún preprocesamiento que llamamos ‘minimización’ de las imágenes del JPEG, reteniendo para cada textura solamente los puntos necesarios para la visualización tridimensional; en otras palabras, las partes de la textura dada que no se usan nunca en la etapa de representación tridimensional pueden comprimirse tan groseramente como queramos).

2.4.2. Compresión del campo Octárbol

La compresión de Octárbol es la parte más importante de la compresión de la ImagenOctárbol, ya que trata de la compresión de una representación ya muy compacta de árbol binario sin enlaces. Sin embargo, en nuestros experimentos, el procedimiento explicado más adelante redujo el volumen de esta estructura hasta alrededor de la mitad

del original. En la versión animada de ImagenOctárbol, el campo octárbol se comprime por separado para cada trama tridimensional.

2.4.2.1 Modelo del contexto

La compresión es llevada a cabo por una variante de la codificación aritmética adaptable (implementada como un 'codificador de distancias') que hace uso explícito de la naturaleza geométrica de los datos. El Octárbol es un flujo de octetos. Cada octeto representa un nodo (es decir, un subcubo) del árbol, en el cual sus bits indican la ocupación del subcubo después de la subdivisión interna. El patrón de bits se llama el patrón de rellenado del nodo. El algoritmo de compresión descrito procesa los octetos uno por uno, de la siguiente manera.

* Se determina un contexto para el octeto actual.

* Se extrae la 'probabilidad' (frecuencia normalizada) de ocurrencia del octeto actual en este contexto de la 'tabla de probabilidades' (PT) correspondiente al contexto.

* El valor de probabilidad se suministra al codificador de distancias.

* La PT actual se actualiza añadiendo 1 a la frecuencia de ocurrencia del octeto actual en el contexto actual (y, si es necesario, se renormaliza después; véanse los detalles más adelante).

Por tanto, la codificación es el proceso de construir y actualizar las PT según el modelo de contexto. En los esquemas aritméticos adaptables de codificación basados en el contexto (tales como la 'Predicción con Coincidencia Parcial'), el contexto de un símbolo es usualmente una cadena de varios símbolos precedentes. Sin embargo, en nuestro caso, la eficacia de la compresión aumenta explotando la estructura del octárbol y la naturaleza geométrica de los datos. El enfoque descrito se basa en las dos ideas que son aparentemente nuevas en el problema de la compresión del octárbol.

A. Para el nodo actual, el contexto es bien su nodo padre, o bien el par {nodo padre, posición actual del nodo en el nodo padre};

B. Se supone que la 'probabilidad' de la ocurrencia del nodo dado en la ubicación geométrica específica en el nodo padre específico es invariante con respecto a un cierto conjunto de transformadas ortogonales (tales como las rotaciones o simetrías).

La hipótesis 'B' se ilustra en la FIG. 6, para la transformada R, que es la rotación en -90º en el plano x-z. La noción básica detrás de 'B' es la observación de que la probabilidad de ocurrencia de un tipo específico de nodo hijo en un tipo específico de nodo padre debería depender solamente de su posición relativa. Esta hipótesis se confirma en nuestros experimentos, por el análisis de las tablas de probabilidades. Nos permite usar un contexto más complejo sin tener demasiadas tablas de probabilidades. Esto, a su vez, ayuda a lograr resultados bastante buenos en términos del tamaño y la velocidad de los datos. Obsérvese que cuantos más contextos se usen, más precisa es la probabilidad estimada y, por tanto, más compacto es el código.

Introduzcamos el conjunto de transformadas para las cuales supondremos la invariancia de las distribuciones de probabilidad. A fin de aplicarse a nuestra situación, tales transformadas deberían preservar el cubo circundante. Consideremos un conjunto G de las transformadas ortogonales en el espacio euclidiano, que se obtienen por todas las composiciones, en cualquier número y orden, de las 3 transformadas básicas (generadores) m_1, m_2 y m_3, dadas por

$$\mathbf{m}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{m}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{m}_3 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

donde m_1 y m_2 son reflejos sobre los planos x = y e y = z, respectivamente, y m_3 es el reflejo sobre el plano x = 0. Uno de los resultados clásicos de la teoría de grupos generados por reflejos afirma que G contiene 48 transformadas ortogonales distintas y es, en cierto sentido, el máximo grupo de transformadas ortogonales que llevan al cubo a sí mismo (el llamado grupo de Coxeter). Por ejemplo, la rotación R en la FIG. 6 se expresa, mediante los generadores, como

$$\mathbf{R = m_3 \cdot m_2 \cdot m_1 \cdot m_2,}$$

donde '·' es la multiplicación matricial.

La transformada desde G, aplicada a un nodo de octárbol, produce un nodo con un patrón distinto de rellenado de subcubos. Esto nos permite categorizar los nodos según el patrón de rellenado de sus subcubos. Usando el lenguaje de la

teoría de grupos [5], decimos que G actúa sobre el conjunto de todos los patrones de rellenado de los nodos de octárbol. Los cálculos muestran que existen 22 clases distintas (también llamadas órbitas en la teoría de grupos), en las cuales, por definición, dos nodos pertenecen a la misma clase si, y sólo si, están conectados por una transformada desde G. El número de elementos en una clase varía entre 1 a 24, y es siempre un divisor de 48.

La consecuencia práctica de la hipótesis 'B' es que la tabla de probabilidades depende no del nodo padre en sí mismo, sino solamente de la clase a la cual pertenece el nodo padre. Obsérvese que habría 256 tablas para un contexto basado en padres y 256x8 = 2048 tablas adicionales para el contexto basado en la posición de padre e hijo en el caso anterior, mientras que sólo necesitamos 22 tablas para el contexto basado en la clase del padre, más 22x8 = 176 tablas en el último caso. Por lo tanto, es posible usar un contexto equivalentemente complejo con un número relativamente pequeño de tablas de probabilidades. La PT construida tendría la forma que se muestra en la Tabla 8.

Tabla 8. Enumeración de tablas de probabilidades.

Identificador de las PT	0	1	...	255	Descripción del contexto
0	P0,0	P0,1	...	P0,255	Contexto-0: independiente del contexto
1..22 (22)	Pi,0	Pi,1	...	Pi,255	Contexto-1: {clase del nodo padre}
23...198 (176)	Pj,0	Pj,1	...	Pj,255	Contexto-2: {clase del nodo padre, posición del nodo actual}

2.4.2.2. Proceso de codificación

Para hacer que las estadísticas para las tablas de probabilidades sean más precisas, se recolectan de distintas maneras en tres etapas del proceso de codificación.

* En la primera etapa no usamos contextos en absoluto, aceptando el 'modelo de contexto-0', y mantenemos una única tabla de probabilidades con 256 entradas, a partir de la distribución uniforme;

* En cuanto los primeros 512 nodos (es un número hallado empíricamente) están codificados, conmutamos al 'modelo de contexto-1', usando el nodo padre como un contexto. En el momento de la conmutación, la PT del contexto-0 se copia a las PT para todos los 22 contextos.

* Después de que los próximos 2048 nodos (otro valor heurístico) estén codificados, conmutamos al 'modelo de contexto-2'. En este momento, las PT de contexto-1 de los patrones padre se copian a las PT para cada posición en el mismo patrón padre.

El punto clave del algoritmo es la determinación del contexto y la probabilidad para el octeto actual. Esto se implementa de la siguiente manera. En cada clase fijamos un único elemento, que se llama el 'elemento estándar'. Almacenamos una tabla de mapas de clase (CMT) que indica la clase a la cual pertenece cada uno de los posibles 256 nodos, y la transformada precalculada de G que lleva a este nodo específico al elemento estándar de su clase. Así, a fin de determinar la probabilidad del nodo actual N, realizamos las siguientes etapas:

* Mirar al padre P del nodo actual;

* Extraer la clase de la CMT a la cual pertenece P, y la transformada T que lleva a P al nodo estándar de la clase. Sea c el número de clase;

* Aplicar T a P, y hallar la posición p del hijo en el nodo estándar al cual está asociado el nodo actual N;

* Aplicar T a N. Luego, el patrón TN de rellenado recién obtenido está en la posición p en el nodo estándar de la clase c.

* Extraer la probabilidad requerida de la entrada TN de la tabla de probabilidades correspondiente a la combinación de clase y posición (c, p).

Para el modelo de contexto-1, las etapas anteriores se modifican en una forma obvia. No hace falta decir que todas las transformaciones están precalculadas, e implementadas en una tabla de búsqueda.

Obsérvese que en la etapa de descodificación del nodo N, su padre P ya está descodificado y, por tanto, la transformada T es conocida. Todas las etapas en la fase de descodificación son absolutamente similares a las correspondientes etapas de codificación.

Finalmente, esbocemos el proceso de actualización de probabilidades. Sea P una tabla de probabilidades para algún contexto. Sea P(N) la entrada de P correspondiente a la probabilidad de ocurrencia del nodo N en este contexto. En nuestra implementación, P(N) es un entero y, después de cada ocurrencia de N, P(N) se actualiza como:

P(N)=P(N)+A,

donde A es un parámetro incremental entero que varía habitualmente entre 1 a 4 para distintos modelos de contexto. Sea S(P) la suma de todas las entradas en P: Entonces, la 'probabilidad' de N que se suministra al codificador aritmético (codificador de distancias en nuestro caso) se calcula como P/(N) / S(P). En cuando S(P) llega a un valor de umbral 2^{16}, todas las entradas se renormalizan: a fin de evitar la ocurrencia de valores nulos en P, las entradas iguales a 1 se dejan intactas, mientras que las otras se dividen entre 2.

2.4.2.3 Codificación de los 'nodos de cámara'

El flujo de símbolos que determinan los números de textura (cámara) para cada voxel se comprime usando su propia tabla de probabilidades. En los términos usados anteriormente, tiene un único contexto. Las entradas de la PT se actualizan con un incremento mayor que el de las entradas para los nodos de octárbol; en el resto, no hay ninguna diferencia con la codificación de símbolos de nodo.

2.5. Resultados de la compresión y representación de TBVO

Las FIGs. 15, 17, 18 y 19 son los resultados de la compresión de TBVO. En la FIG. 16, se ilustran las imágenes despojadas de los modelos "Ángel" y "Morton". El tamaño comprimido se compara con el BVO comprimido: en la tercera columna, el número entre corchetes es el volumen de geometría comprimido, mientras que el primer número es el volumen total del modelo comprimido basado en TBVO (es decir, se tienen en cuenta las texturas). Como medida de la distorsión visual, se calculó la PSNR (Razón de Señal a Ruido de Imagen) para estimar la diferencia de color después de la transformada LDI->7(T)BVO->LDI. El tamaño del modelo comprimido es el tamaño de todas las texturas (almacenadas como JPEG minimizadas, véase 0), más el tamaño de la geometría comprimida. En el caso de TBVO, la geometría comprimida incluye también la información de cámara. La PSNR del TBVO mejora significativamente en comparación con el BVO.

El TBVO logra una representación más rápida que el BVO. Para el modelo "Ángel", la velocidad de tramas de TBVO-12 es 10,8 tramas por segundo, mientras que la de BVO es 7,5. Para el modelo "Morton", la de TBVO-12 es 3,0 tramas por segundo, mientras que la de BVO es 2,1 (en un Celeron a 850 MHz). Por otra parte, se observa que la representación se acelera mucho más en el TBVO animado. Para el modelo "Dragón", la velocidad de tramas de TBVO-12 es de 73 tramas por segundo, mientras que la de BVO es de 29 tramas por segundo (en un Pentium IV a 1,8 GHz).

Un formato de TBVO brinda gran flexibilidad. Por ejemplo, se ilustran 2 maneras de usar 12 cámaras en la FIG. 6: TBVO-12 y TBVO-(6+6). TBVO-12 usa 6 cámaras de BVO (caras de cubo) más 6 imágenes tomadas desde el centro del cubo, y paralelas a las caras. La configuración (6+6) usa 6 cámaras de BVO, y luego elimina ('pela') todos los voxels visibles por estas cámaras y 'fotografía' las partes que se tornaron visibles por las mismas 6 cámaras. Ejemplos de tales imágenes se muestran en la FIG. 16.

Obsérvese la drástica diferencia en calidad (subjetiva, y el valor de la PSNR) entre los modelos del Ángel de BVO y TBVO-6. Aunque se usan las mismas ubicaciones de cámara, TBVO nos permite asignar números de cámara a todos los voxels, incluso aquellos invisibles a todas las cámaras. Estos números se escogen a fin de coincidir óptimamente con los colores originales (es decir, para cada punto se selecciona la mejor coincidencia de color en todas las 'imágenes de cámara', independientemente de la visibilidad directa. En el caso del Ángel da un gran resultado).

Obsérvese también la muy modesta diferencia de volumen de 'geometría' (es decir, BVO + cámaras) entre los casos 6 y 12 de cámara. En verdad, las cámaras adicionales cubren, habitualmente, regiones pequeñas y, por tanto, sus identificadores son raros, y sus texturas son ralas (y bien comprimidas). Todo esto se aplica no solamente al 'Ángel', sino también a 'Morton', 'Palmera512' y 'robots512'.

2.6. Especificación del nodo

ImagenOctárbol {

campo	SFInt32	resoluciónoctárbol 256	
campo	MFInt32	octárbol	[]#%q=13,8
campo	MFInt32	IdentificadorCámara	[]#%q=13,8
campo	MFNode	imágenesoctárbol	[]

}

El nodo ImagenOctárbol define una estructura de TBVO, en la cual existen una estructura de octárbol, la correspondiente formación de índices de cámara y un conjunto de imágenesoctárbol.

El campo imágenesoctárbol especifica un conjunto de nodos de ImagenConProfundidad con TexturaSimple para el campo diTextura; el campo profundidad no se usa en estos nodos de TexturaSimple. El campo ortográfico debe ser TRUE para los nodos de ImagenConProfundidad. Para cada una de las TexturasSimples, el campo textura almacena la información de color del objeto, o parte de la vista del objeto (por ejemplo, su sección transversal por un plano de cámara) según lo obtenido por la cámara ortográfica cuya posición y orientación se especifican en los correspondientes campos de ImagenConProfundidad. Partes del objeto, correspondientes a cada cámara, se asignan en la etapa de la construcción del modelo. La partición del objeto, usando los valores de los campos de posición, orientación y textura, se efectúa a fin de minimizar el número de cámaras (o, equivalentemente, de las imágenesoctárbol implicadas), para incluir a la vez todas las partes del objeto potencialmente visibles desde una posición arbitraria escogida. Los campos de orientación deben satisfacer la condición: el vector de vista de cámara tiene solamente una componente no nula (es decir, es perpendicular a una de las caras del cubo circundante). Además, los lados de la imagen de TexturaSimple deben ser paralelos a los correspondientes lados del cubo circundante.

El campo octárbol describe completamente la geometría del objeto. La geometría se representa como un conjunto de voxels, que constituye el objeto dado. Un octárbol es una estructura de datos arbolada, en la cual cada nodo se representa por un octeto. Un 1 en el i-ésimo bit de este octeto significa que existen los nodos hijos para el i-ésimo hijo de ese nodo interno; mientras que un 0 significa que no existen. El orden de un nodo interno se muestra en la FIG. 14 (b). El tamaño del cubo circundante del octárbol total es 1x1x1, y el centro del cubo del octárbol será el origen (0, 0, 0) del sistema de coordenadas locales.

El campo IdentificadorCámara contiene una formación de índices de cámaras asignados a los voxels. En la etapa de representación, el color atribuido a una hoja terminal del octárbol se determina proyectando ortográficamente la hoja sobre una de las imágenesoctárbol, con un índice específico. Los índices se almacenan en forma de octárbol: si puede usarse una cámara específica para todas las hojas contenidas en un nodo específico, el nodo que contiene el índice de la cámara se emite al flujo; en caso contrario, se emite el nodo que contiene un código fijo de 'subdivisión adicional', lo que significa que el índice de cámara se especificará por separado para los subnodos hijos del nodo actual (de la misma forma recursiva). Si el IdentificadorCámara está vacío, entonces los índices de cámara se determinan durante la etapa de representación (como en el caso de BVO).

El campo resoluciónoctárbol especifica el máximo número admisible de hojas terminales del octárbol a lo largo de un lado del cubo circundante. El nivel del octárbol puede determinarse a partir de resoluciónoctárbol, usando la siguiente ecuación:

$$niveloctárbol = \lceil \log_2 (resoluciónoctárbol) \rceil$$

2.7. Especificación del flujo de bits

2.7.1. Compresión del octárbol

2.7.1.1. Panorama general

El nodo ImagenOctárbol en la Representación Basada en Imagen con Profundidad define la estructura del octárbol y sus texturas proyectadas. Cada textura, almacenada en la formación de ImágenesOctárbol, se define a través del nodo ImagenConProfundidad con TexturaSimple. Los otros campos del nodo ImagenOctárbol pueden comprimirse por compresión del octárbol.

2.7.1.2. Octárbol

2.7.1.2.1. Sintaxis

```
class Octárbol ()
  CabeceraOctárbol ();
  aligned bit (32)* next;
  while (next == 0x000001C8)
  {
    aligned bit (32) código_inicio_trama_octárbol;
```

```
    TramaOctárbol (NivelOctárbol)
    aligned bit (32) * next;
  }
}
```

2.7.1.2.2. Semántica

El flujo comprimido del octárbol contiene una cabecera de octárbol y una o más tramas de octárbol, cada una precedida por el código_inicio_trama_octárbol. El valor de código_inicio_trama_octárbol es siempre 0x000001C8. Este valor se detecta por el análisis sintáctico adelantado (next) del flujo.

2.7.1.3. CabeceraOctárbol

2.7.1.3.1. Sintaxis

```
class CabeceraOctárbol ()
{unsigned int (5) BitsResoluciónOctárbol;
unsigned int (BitsResoluciónOctárbol) ResoluciónOctárbol;
int niveloctárbol = ceil (log (ResoluciónOctárbol) / log (2));
unsigned int (3) NúmBitsTextura;
unsigned int (NumBitsTextura) NúmDeTexturas;
}
```

2.7.1.3.2. Semántica

Esta clase lee la información de cabecera para la compresión del octárbol.

La ResoluciónOctárbol, cuya longitud está descrita por BitsResoluciónOctárbol, contiene el valor del campo resoluciónoctárbol del nodo ImagenOctárbol. Este valor se usa para obtener el nivel del octárbol.

El NúmDeTexturas, que tiene un largo de NúmBitsTextura, describe el número de texturas (o cámaras) usadas en el nodo ImagenOctárbol. Este valor se usa para la codificación aritmética del Identificador de cámara para cada nodo del octárbol. Si el valor de NúmBitsTextura es 0, entonces no se codifican los símbolos de textura, fijando el valor de TexturaAct del nodo raíz en 255.

2.7.1.4. TramaOctárbol

2.7.1.4.1. Sintaxis

```
class TramaOctárbol (int NivelOctárbol)
  for (int NivelAct=0; NivelAct < NivelOctárbol; NivelAct++0
    for (int ÍndiceNodo=0; ÍndiceNodo < nNodosEnNivelAct; ÍndiceNodo++)
{
    int SímNodo = SímboloDescodificadorAritmético (IdentificadorContexto);
    if (TexturaAct == 0)
    {
      TexturaAct = SímboloDescodificadorAritmético (IdentificadorContextoTextura);
    }
  for (int ÍndiceNodo=0; ÍndiceNodo < nNodosEnNivelAct; ÍndiceNodo++)
    if (TexturaAct == 0)
```

TexturaAct = SímboloDescodificadorAritmético (IdentificadorContextoTextura);

{

2.7.1.4.2. Semántica

Esta clase lee una única trama de octárbol en un orden de recorrido de amplitud primero. A partir del 1er nodo en el nivel 0, después de leer cada nodo en el nivel actual, se conoce el número de nodos en el próximo nivel contando todos los 1 en cada símbolo de nodo. En el próximo nivel, se leerá ese número de nodos (nNodosEnNivelAct) del flujo.

Para la descodificación de cada nodo, se da un IdentificadorContexto adecuado, según lo descrito en la cláusula 2.7.1.6.

Si el Identificador de textura (o cámara) para el nodo actual (TexturaAct) no está definido por el nodo padre, entonces el Identificador de textura también se lee del flujo, usando el contexto para el Identificador de textura, definido por IdentificadorContextoTextura. Si se extrae un valor no nulo (el Identificador de textura está definido), entonces este valor también se aplicará a todos los nodos hijos en los siguientes niveles. Después de descodificar cada nodo, el IdentificadorTextura se asignará a los nodos de hoja terminal del octárbol que aún no tengan asignado el valor de IdentificadorTextura.

2.7.1.5. Descodificación Aritmética Adaptable

En esta sección se describe el codificador aritmético adaptable usado en la compresión del octárbol, usando la descripción sintáctica de estilo C++. descodificador_aa () es la función que descodifica un símbolo, usando un modelo especificado por toda la formación frec_acumul[] y PCT es una formación de tablas de contexto de probabilidad, según lo descrito en la cláusula 2.7.1.6.

```
int SímboloDescodificadorAritmético (int IdentificadorContexto)
  unsigned int MAXCUM = 1 <<13;
  unsigned int MAXCUMTextura = 256;
  int *p, todosím, maxcum;
  if (IdentificadorContexto != IdentificadorContextoTextura)
     p = PCT [IdentificadorContexto];
     todosím = 256;
     maxcum = MAXCUM;
  else
    p = PCTTextura;
    todosím = númDeTexturas;
    maxcum = MAXCUMTextura; }
  int frec_acumul [todosím];
  int cum = 0;
  for (int i=todosím-1; i >=0; i--)
   { cum += p[i];
     frec_acumul[i] = cum;
  if (cum > maxcum)
   { cum = 0;
  for (int i=todosím-1; i>=0; i-)
    {  PCT[IdentificadorContexto] [i] = (PCT[IdentificadorContexto] [i] +1)2;
     cum += PCT[IdentificadorContexto] [i];
```

FrecAcumul [i] = cum; }

devolver descodificador_aa (frec_acumul);

{

2.7.1.6. Proceso de descodificación

La estructura general del proceso de descodificación se describe en la cláusula 0 (véase también la descripción anterior del proceso de codificación). Muestra cómo se obtienen los nodos de TBVO del flujo de bits que constituyen el modelo de TBVO aritméticamente codificado (comprimido).

En cada etapa del proceso de descodificación debemos actualizar el número de contexto (es decir, el índice de la tabla de probabilidades que usamos), y la tabla de probabilidades en sí. Llamamos modelo Probabilístico a la unión de todas las tablas de probabilidades (formaciones enteras). El j-ésimo elemento de la i-ésima tabla de probabilidades, dividido entre la suma de sus elementos, estima la probabilidad de ocurrencia del j-ésimo símbolo en el i-ésimo contexto.

El proceso de actualización de la tabla de probabilidades es el siguiente. En el comienzo, las tablas de probabilidades están inicializadas de modo tal que todas las entradas sean iguales a 1. Antes de descodificar un símbolo, debe escogerse el número de contexto (IdentificadorContexto). El IdentificadorContexto se determina a partir de datos anteriormente descodificados, según lo indicado en 0 y 0 más adelante. Cuando se obtiene el IdentificadorContexto, el símbolo se descodifica usando el descodificador aritmético binario. Después de eso, se actualiza la tabla de probabilidades, añadiendo una etapa de adaptación a la frecuencia del símbolo descodificado. Si la suma total (acumulativa) de elementos de tabla se torna mayor que el umbral acumulativo, entonces se realiza la normalización (véase 2.7.1.5.1).

2.7.1.6.1. Modelación del contexto del símbolo de textura

El símbolo de textura se modela con sólo un contexto. Esto significa que sólo se usa una tabla de probabilidades. El tamaño de esta tabla es igual al número NúmDeTexturas más uno. En el comienzo, esta tabla se inicializa con valores todos iguales a 1. El máximo valor admisible de entrada se fija en 256. La etapa adaptable se fija en 32. Esta combinación de valores paramétricos permite la adaptación a un flujo sumamente variable de los números de textura.

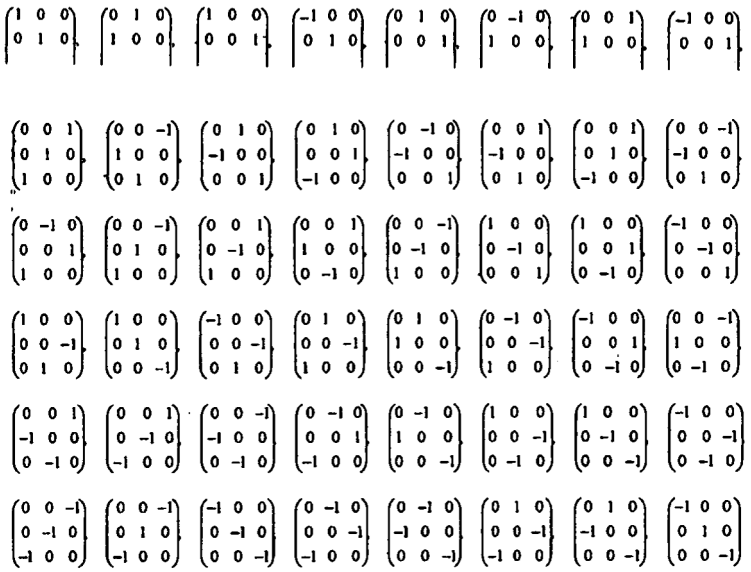
2.7.1.6.2. Modelación del contexto del símbolo de nodo

Hay 256 símbolos distintos de nodo, representando cada símbolo a una formación binaria de voxels de dimensiones 2x2x2. Puede aplicarse una transformación ortogonal tridimensional a estas formaciones, transformando los correspondientes símbolos entre sí.

Consideremos un conjunto de 48 transformadas ortogonales fijas, es decir, rotaciones en 90*n (n=0,1,2,3) grados alrededor de los ejes de coordenadas, y simetrías. Sus matrices se dan a continuación, en el orden de sus números:

Transformadas Ortogonales [48]=

{



}

Hay 22 conjuntos de símbolos – llamados clases -, de modo tal que 2 símbolos están conectados por una transformada tal si y sólo si pertenecen a la misma clase. El procedimiento de codificación construye las PCT de la siguiente manera: el IdentificadorContexto de un símbolo es igual, bien al número de clase a la cual pertenece su padre, o bien a un número combinado (clase del padre, posición del nodo actual en el nodo padre). Esto permite una gran reducción en el número de contextos, reduciendo el tiempo necesario para obtener estadísticas significativas.

Para cada clase, se determina un único símbolo base (véase la Tabla II y, para cada símbolo, se precalcula la transformada ortogonal que la lleva al símbolo base de su clase (en el proceso efectivo de codificación / descodificación, se usa la tabla de búsqueda). Después de que se determina el IdentificadorContexto para un símbolo, se aplica la transformada, inversa (es decir, matriz traspuesta) de la que lleva a su padre al elemento base. En la Tabla 9, se dan los contextos y las correspondientes transformadas directas para cada símbolo.

Tabla 9. Ejemplo de símbolo base para cada clase

Clase	Ejemplo de símbolo base	Orden de la clase (Número de elementos)
0	0	1
1	1	8
2	3	12
3	6	12
4	7	23
5	15	6
6	22	8
7	23	8
8	24	4
9	25	24

10	27	24
11	30	24
12	31	24
13	60	6
14	61	24
15	63	12
16	105	2
17	107	8
18	111	12
19	126	4
20	127	8
21	255	1

El modelo de contexto depende del número N de símbolos ya descodificados:

Para N < 512 hay sólo un contexto. La tabla de probabilidades se inicializa con valores todos 1. El número de símbolos en la tabla de probabilidades es 256. La etapa de adaptación es 2. La máxima frecuencia acumulativa es 8.192.

Para 512 $\leq$ N < 2.560 (=2.048+512), se usa el modelo de contexto-1 (en el sentido de que el número de contexto es parámetro único, el número de la clase). Este modelo usa 22 PCT. El IdentificadorContexto es el número de la clase a la cual pertenece el padre del nodo descodificado. Este número siempre puede determinarse a partir de la tabla de búsqueda (véase la Tabla III), porque el padre se descodifica antes que el hijo. Cada una de las 22 PCT es inicializada por la PCT de la etapa anterior. El número de símbolos en cada tabla de probabilidades es 256. La etapa de adaptación es 3. La máxima frecuencia acumulativa es también 8.192. Después de que se descodifica el símbolo, se transforma usando la transformada ortogonal inversa definida anteriormente. El número de transformada ortogonal puede hallarse en la Tabla III, con el Identificador de Símbolo de Nodo igual al padre del símbolo del nodo actual.

Cuando se descodifican 2.560 símbolos, el descodificador conmuta al contexto-2 (en el sentido de que el número de contexto está ahora compuesto por los dos parámetros, según se explica más adelante). Este modelo usa 176 (=22*8, es decir, 22 clases por 8 posiciones) PCT. El IdentificadorContexto aquí depende de la clase del padre y de la posición del nodo actual en el nodo padre. Las tablas iniciales de probabilidades para este modelo dependen sólo de su contexto, pero no de la posición: para todas las 8 posiciones, PCT es un clon de la PCT obtenida para la clase dada en la etapa anterior. El número de símbolos en cada tabla de probabilidades es 256. La etapa de adaptación es 4. La máxima frecuencia acumulativa también es 8.192.

Después de que el símbolo es descodificado, también se transforma usando la transformada ortogonal inversa (a la dada en la Tabla III), como ocurre en el modelo anterior.

Puede obtenerse fácilmente la geometría de los elementos base para cada clase, usando la Tabla 10. Los elementos base son exactamente los símbolos para los cuales el Identificador de Transformada es 0 (el número 0 se asigna a la transformada idéntica).

Tabla 10. Tabla de búsqueda conjunta para el símbolo de nodo, su número de clase y la transformada ortogonal que lleva el símbolo al elemento base fijo de esta clase

Identificador de Símbolo de Nodo	Identificador Clase	Identificador de Transformada Ortogonal	Identificador Símbolo de Nodo	Identificador Clase	Identificador de Transformada Ortogonal	Identificador Símbolo De Nodo	Identificador Clase	Identificador de Transformada Ortogonal
0	0	0	85	5	6	170	5	9
1	1	0	86	11	6	171	12	9
2	1	3	87	12	6	172	10	20
3	2	0	88	9	37	173	14	12
4	1	10	89	11	13	174	12	15
5	2	1	90	13	1	175	15	5
6	3	0	91	14	1	176	4	36
7	4	0	92	10	18	177	10	25
8	1	12	93	12	13	178	7	30
9	3	3	94	14	10	179	12	30
10	2	5	95	15	1	180	11	38
11	4	3	96	3	25	181	14	19
12	2	21	97	6	11	182	17	16
13	4	10	98	9	36	183	18	7
14	4	12	99	11	11	184	10	31
15	5	0	100	9	38	185	14	35
16	1	11	101	11	14	186	12	31
17	2	4	102	13	4	187	15	16
18	3	2	103	14	4	188	14	39
19	4	2	104	6	34	189	19	3
20	3	6	105	16	0	190	18	9
21	4	6	106	11	34	191	20	3
22	6	0	107	17	0	192	2	37
23	7	0	108	11	39	193	9	32
24	8	0	109	17	1	194	9	34
25	9	0	110	14	20	195	13	21
26	9	7	111	18	0	196	4	37
27	10	0	112	4	25	197	10	27
28	9	13	113	7	11	198	11	26
29	10	1	114	10	22	199	14	21
30	11	0	115	12	11	200	4	39
31	12	0	116	10	19	201	11	24

(cont.)								
32	1	30	117	12	14	202	10	29
33	3	7	118	14	11	203	14	23
34	2	16	119	15	4	204	5	24
35	4	7	120	11	42	205	12	24
36	8	2	121	17	4	206	12	26
37	9	2	122	14	31	207	15	21
38	9	3	123	18	2	208	4	38
39	10	2	124	14	37	209	10	28
40	3	9	125	18	6	210	11	36
41	6	3	126	19	0	211	14	22
42	4	9	127	20	0	212	7	32
43	7	3	128	1	34	213	12	32
44	9	15	129	8	9	214	17	18
45	11	3	130	3	15	215	18	13
46	10	5	131	9	9	216	10	37
47	12	3	132	3	26	217	14	33
48	2	22	133	9	24	218	14	34
49	4	11	134	6	12	219	19	10
50	4	30	135	11	12	220	12	37
51	5	2	136	2	20	221	15	18
52	9	14	137	9	12	222	18	24
53	10	4	138	4	15	223	20	10
54	11	2	139	10	9	224	4	42
55	12	2	140	4	26	225	11	25
56	9	31	141	10	23	226	10	34
57	11	7	142	7	12	227	14	30
58	10	16	143	12	12	228	10	38
59	12	7	144	3	36	229	14	32
60	13	0	145	9	25	230	14	40
61	14	0	146	6	30	231	19	11
62	14	3	147	11	30	232	7	34
63	15	0	148	6	32	233	17	20
64	1	32	149	11	32	234	12	34
65	3	13	150	16	3	235	18	15
66	8	6	151	17	3	236	12	39

(cont.)								
67	9	6	152	9	42	237	18	26
68	2	18	153	13	16	238	15	20
69	4	13	154	11	31	239	20	12
70	9	10	155	14	16	240	5	25
71	10	6	156	11	37	241	12	25
72	3	24	157	14	18	242	12	36
73	6	10	158	17	5	243	15	22
74	9	26	159	18	3	244	12	38
75	11	10	160	2	31	245	15	19
76	4	24	161	9	30	246	18	25
77	7	10	162	4	31	247	20	11
78	10	21	163	10	17	248	12	42
79	12	10	164	9	39	249	18	36
80	2	19	165	13	5	250	15	31
81	4	14	166	11	15	251	20	30
82	9	11	167	14	5	252	15	37
83	10	8	168	4	34	253	20	32
84	4	32	169	11	9	254	20	34
						255	21	0

De aquí en adelante, se describirán en detalle la especificación del nodo de MPEG-4 y las técnicas de compresión de los formatos de imagen de octárbol usados en el aparato y procedimiento de representación tridimensional basado en imágenes con profundidad, según la presente invención.

Esta invención describe una familia de estructuras de datos, representaciones basadas en imágenes con profundidad (DIBR), que proporcionan representaciones efectivas y eficaces, basadas mayormente en imágenes y mapas de profundidades, utilizando completamente las ventajas descritas anteriormente. Caractericemos brevemente los principales formatos de la DIBR: TexturaSimple, TexturaDePunto e ImagenOctárbol.

La FIG. 20 es un diagrama de un ejemplo de la imagen de textura y el mapa de profundidades, y la FIG. 21 es un diagrama de un ejemplo de Imagen de Profundidad en Capas (LDI). (a) muestra la proyección del objeto y (b) muestra los píxeles en capas.

La TexturaSimple es una estructura de datos que consiste en una imagen, el correspondiente mapa de profundidades, y la descripción de la cámara (su posición, orientación y tipo, ortogonal o en perspectiva). Las capacidades de representación de una única TexturaSimple se restringen a objetos como la fachada de un edificio: una imagen frontal con mapa de profundidades, que permite la reconstrucción de vistas de la fachada en una gama significativa de ángulos. Sin embargo, la colección de las Texturas Simples producidas por cámaras adecuadamente situadas permite la representación del edificio entero – en caso de que las imágenes de referencia cubran todas las partes potencialmente visibles de la superficie del edificio. Por supuesto, lo mismo vale para los árboles, figuras humanas, automóviles, etc. Además, la unión de TexturasSimples brinda medios bastante naturales para manipular datos animados tridimensionales. En este caso, las imágenes de referencia son reemplazadas por flujos de vídeo de referencia. Los mapas de profundidades para cada trama tridimensional pueden representarse bien por valores de canal alfa de estos flujos de vídeo, o bien por flujos distintos de vídeo de escalas de grises. En este tipo de representación, las imágenes pueden almacenarse en formatos comprimidos con pérdida como, digamos, JPEG. Esto reduce significativamente el volumen de la información de color, especialmente en el caso animado. Sin embargo, la información de geometría (mapas de profundidades) debería

comprimirse sin pérdidas, lo que afecta a la reducción general en el almacenamiento.

Para los objetos de forma compleja, a veces es difícil cubrir toda la superficie visible con un número razonable de imágenes de referencia. La representación preferible para tales casos podría ser la TexturaDePunto. Este formato también almacena una imagen de referencia y un mapa de profundidades, pero en este caso ambos están multivaluados: para cada línea de visión proporcionada por la cámara (ortográfica o en perspectiva), se almacenan el color y la distancia para cada intersección de la línea con el objeto. El número de intersecciones puede variar de línea a línea. La unión de varias TexturasDePunto proporciona una representación muy detallada, incluso para objetos complejos. Pero el formato carece mayormente de la regularidad bidimensional de la TexturaSimple y, por tanto, no tiene ninguna forma comprimida natural basada en imágenes. Por el mismo motivo, se usa sólo para objetos fijos.

El formato de ImagenOctárbol ocupa una posición intermedia entre la TexturaSimple 'mayormente bidimensional' y la TexturaDePunto 'mayormente tridimensional': almacena la geometría del objeto en la representación volumétrica estructurada como octárbol (voxels jerárquicamente organizados de la usual subdivisión binaria del cubo circundante), mientras que el componente de color está representado por un conjunto de imágenes. Este formato contiene también una estructura adicional de datos similares a un octárbol, que almacena, para cada voxel de hoja terminal, el índice de una imagen de referencia que contiene su color. En la etapa de representación de la ImagenOctárbol, el color del voxel de hoja terminal se determina proyectándolo ortográficamente sobre la correspondiente imagen de referencia. Hemos desarrollado un procedimiento de compresión muy eficaz para la parte de geometría de la ImagenOctárbol. Es una variante de la codificación aritmética adaptable basada en el contexto, donde los contextos se construyen con el uso explícito de la naturaleza geométrica de los datos. El uso de la compresión, junto con las imágenes de referencia comprimidas con pérdidas, hace de ImagenOctárbol una representación muy eficaz en términos de espacio. Como la TexturaSimple, la ImagenOctárbol tiene una versión animada: flujos de vídeo de referencia en lugar de imágenes de referencia, más dos flujos adicionales de octárboles que representan la geometría y la correspondencia entre voxel e imagen para cada trama tridimensional. Una característica muy útil de un formato de ImagenOctárbol es su capacidad implícita de asociación media.

La familia DIBR ha sido desarrollada para la nueva versión del estándar MPEG-4, y adoptada para su inclusión en la extensión del Entorno de Animación (AFX) de MPEG. AFX proporciona más características mejoradas para entornos sintéticos de MPEG-4, e incluye una colección de herramientas interoperables que producen una arquitectura reutilizable para contenidos animados interactivos (compatibles con el MPEG-4 existente). Cada herramienta de la AFX muestra la compatibilidad con un nodo BIFS, un flujo sintético y un flujo audiovisual. La versión actual de la AFX consiste en descripciones de nivel superior de la animación (p. ej., animación basada en piel y hueso), la representación mejorada (p. ej., la texturización procedural, la asociación de luz y campos), las representaciones compactas (p. ej., NURBS, la representación sólida, las superficies de subdivisión), las animaciones de baja velocidad de bits (p. ej., la compresión de interpolador) y otros, así como nuestra DIBR propuesta.

Los formatos de DIBR se diseñaron a fin de combinar las ventajas de distintas ideas sugeridas anteriormente, proporcionando a un usuario las herramientas flexibles mejor adaptadas para una tarea específica. Por ejemplo, la TexturaSimple y la TexturaDePunto no animadas son casos específicos de los formatos conocidos, mientras que ImagenOctárbol es una representación aparentemente nueva. Pero en el contexto del estándar MPEG-4, los tres formatos básicos de DIBR pueden considerarse como cimientos, y sus combinaciones por medio de construcciones del MPEG-4 no sólo abarcan muchas de las representaciones basadas en imágenes sugeridas en las bibliografías, sino que también brindan un gran potencial para construir tales nuevos formatos.

Ahora se describirá la Representación Basada en Imágenes con Profundidad.

Teniendo en cuenta las ideas esbozadas en la sección anterior, así como algunos de nuestros propios desarrollos, sugerimos el siguiente conjunto de formatos basados en imágenes, para su uso en la AFX del MPEG-4: TexturaSimple, TexturaDePunto, ImagenConProfundidad e ImagenOctárbol. Obsérvese que TexturaSimple e ImagenOctárbol tienen versiones animadas.

TexturaSimple es una única imagen combinada con una imagen con profundidad. Es equivalente a RT, mientras que TexturaDePunto es equivalente a LDI.

En base a la TexturaSimple y la TexturaDePunto como cimientos, podemos construir una gran variedad de representaciones usando construcciones del MPEG-4. Las especificaciones formales se darán más adelante, y aquí describimos el resultado geométricamente.

La estructura de ImagenConProfundidad define bien una TexturaSimple o bien una TexturaDePunto, junto con el recinto circundante, la posición en el espacio y alguna otra información. Un conjunto de ImágenesConProfundidad puede unificarse bajo una única estructura llamada nodo de Transformada, y esto permite la construcción de una gran variedad de representaciones útiles. Las más usadas mayormente son las dos que no tienen un nombre específico de MPEG-4, pero en nuestra práctica las llamamos Textura de Recinto (BT) y Textura Generalizada de Recinto (GBT). BT es una

unión de seis TexturasSimples correspondientes a un cubo circundante de un objeto o una escena, mientras que GBT es una unión arbitraria de cualquier número de TexturasSimples, que proporcionan en conjunto una representación tridimensional coherente. Un ejemplo de BT se da en la FIG. 22, donde se muestran imágenes de referencia, mapas de profundidades y el objeto tridimensional resultante. La BT puede representarse con ayuda de un algoritmo de distorsión incremental, pero usamos un enfoque distinto, aplicable asimismo a la GBT. Un ejemplo de la representación de GBT se muestra en la FIG. 23, donde se usan 21 TexturasSimples para representar un objeto complejo, la palmera.

Debería observarse que los mecanismos de unificación permiten, por ejemplo, el uso de varias LDI con distintas cámaras, para representar el mismo objeto, o puertos del mismo objeto. Por tanto, las estructuras de datos tales como los objetos basados en imágenes, las células del árbol de LDI, las células de la estructura de árbol basado en surfels, son todos casos particulares de este formato, que obviamente ofrece una flexibilidad mucho mayor en la adaptación de la ubicación y resolución de TexturasSimples y TexturasDePunto a la estructura de la escena.

A continuación, se describirá la ImagenOctárbol: Octárbol Volumétrico Binario Texturizado (TBVO).

A fin de utilizar la geometría y la textura de resolución múltiple con una representación más flexible y una representación rápida, desarrollamos la representación de ImagenOctárbol, que se basa en el Octárbol Volumétrico Binario Texturizado (TBVO). El objetivo del TBVO es idear un formato flexible de representación / compresión con visualización rápida de alta calidad. El TBVO consiste en tres componentes principales: el Octárbol Volumétrico Binario (BVO), que representa la geometría, un conjunto de imágenes de referencia e índices de imágenes correspondientes a los nodos del octárbol.

La información geométrica en forma de BVO es un conjunto de voxels binarios (ocupados o vacíos) regularmente distanciados, combinados en células mayores de la manera usual de los octárboles. Esta representación puede obtenerse fácilmente a partir de datos de ImagenConProfundidad, a través de la forma intermedia de 'nube puntual', ya que cada píxel con profundidad define un único punto en el espacio tridimensional. La conversión de la nube puntual en BVO se ilustra en la FIG. 24. Un proceso análogo permite la conversión del modelo poligonal en BVO. La información de textura del BVO puede extraerse de las imágenes de referencia. Una imagen de referencia es una textura de voxels en una posición y orientación de cámara dadas. Por tanto, el BVO en sí, junto con las imágenes de referencia, ya proporciona la representación del modelo. Sin embargo, resultó que una estructura adicional, que almacene el índice de imagen para cada hoja terminal del BVO, permite una visualización mucho más rápida y con mejor calidad.

El principal problema de visualización del BVO es que debemos determinar el correspondiente índice de cámara de cada voxel durante la representación. A este fin, debemos al menos determinar la existencia de una cámara, desde la cual sea visible el voxel. Este procedimiento es muy lento si usamos el enfoque de fuerza bruta. Además de este problema, aún hay algunas dificultades para los voxels que no son visibles desde ninguna cámara, produciendo distorsiones indeseables en la imagen representada.

Una posible solución podría ser almacenar el color explícito para cada voxel.

Sin embargo, en este caso, hemos experimentado algún problema al comprimir la información de color. Es decir, si agrupamos colores de voxel como un formato de imagen y lo comprimimos, la correlación de colores de los voxels vecinos se destruye, por lo que la razón de compresión sería insatisfactoria.

En el TBVO, el problema se resuelve almacenando el índice de (imagen de) cámara para cada voxel. El índice es usualmente el mismo para grandes grupos de voxels, y esto permite el uso de la estructura de octárbol para un almacenamiento económico de la información adicional. Obsérvese que, en promedio, sólo se observó un aumento de volumen del 15%, en comparación con la representación que usa sólo BVO e imágenes de referencia, en los experimentos con nuestros modelos. Su modelación es un poco más compleja, pero admite una manera más flexible de representar objetos de cualquier geometría.

Obsérvese que el TBVO es una representación muy conveniente para representar con ayuda de los splat, porque el tamaño del splat se calcula fácilmente a partir del tamaño del voxel. El color del voxel se determina fácilmente usando las imágenes de referencia y el índice de imagen del voxel

Ahora se describirá el envío de flujos del octárbol volumétrico binario texturizado. Suponemos que 255 cámaras son suficientes, y asignamos hasta 1 octeto para el índice. El flujo del TBVO es un flujo de símbolos. Cada símbolo del TBVO es un símbolo de BVO o un símbolo de Textura. Los símbolos de Textura indican un índice de cámara, que podría ser un número específico o un código de "indefinido".

Sea '?' el código "indefinido" para la descripción posterior. El flujo de BVO se recorre en orden de amplitud primero. Describamos ahora cómo grabar el flujo de TBVO si tenemos el BVO y cada voxel de hoja terminal tiene índice de imagen. Esto debe hacerse en la etapa de modelación. Recorrerá todos los nodos del BVO, incluyendo los nodos de hoja terminal (que no tienen símbolo de BVO) en orden de amplitud primero. En la FIG. 25, se muestra el seudocódigo que completa la grabación del flujo.

Un ejemplo de grabación del flujo de bits del TBVO se muestra en la FIG. 14. Para el árbol de TBVO mostrado en la FIG. 14(a), puede obtenerse un flujo de símbolos según se muestra en la FIG. 14(c), de acuerdo al procedimiento. En este ejemplo, los símbolos de textura se representan en octetos. Sin embargo, en el flujo efectivo, cada símbolo de textura necesitaría 2 bits, porque sólo necesitamos representar tres valores (dos cámaras y el código indefinido).

A continuación se describirá la Animación de DIBR.

Las versiones animadas se definieron para dos de los formatos de DIBR: ImagenConProfundidad, que contiene sólo TexturasSimples, e ImagenOctárbol. El volumen de datos es una de las cuestiones cruciales en la animación tridimensional. Hemos escogido estos formatos específicos porque los flujos de vídeo pueden incorporarse naturalmente en las versiones animadas, proporcionando una significativa reducción de los datos.

Para la ImagenConProfundidad, la animación se realiza reemplazando las imágenes de referencia por TexturasDePelícula del estándar MPEG-4. La compresión de vídeo con pérdida de alta calidad no afecta seriamente a la apariencia de los objetos tridimensionales resultantes. Los mapas de profundidades pueden almacenarse (en modalidad casi sin pérdida) en los canales alfa de los flujos de vídeo de referencia. En la etapa de representación, la trama tridimensional se representa después de que todas las imágenes de referencia y tramas de profundidad están recibidas y descomprimidas.

La animación de la ImagenOctárbol es similar: las imágenes de referencia son reemplazadas por TexturasDePelícula del estándar MPEG-4, y aparece un nuevo flujo del octárbol.

Se definirá ahora la Especificación del Nodo de MPEG-4.

Los formatos de DIBR se describen en detalle en las especificaciones de los nodos de la AFX del MPEG-4. La ImagenConProfundidad contiene campos que determinan los parámetros de la pirámide truncada a la vista, bien para la TexturaSimple o bien para la TexturaDePunto. El nodo de ImagenOctárbol representa el objeto en la forma de una geometría definida por TBVO y un conjunto de formatos de imagen de referencia. La información dependiente de la escena se almacena en campos especiales de las estructuras de datos de la DIBR, permitiendo la correcta interacción de los objetos de DIBR con el resto de la escena. La definición de los nodos de DIBR se muestra en la FIG. 26.

La FIG. 27 ilustra el diseño especial de la ImagenConProfundidad, en la cual se muestra el significado de cada campo. Obsérvese que el nodo de ImagenConProfundidad define un único objeto de DIBR. Cuando múltiples nodos de ImagenConProfundidad están relacionados entre sí, se procesan como un grupo y, así, deberían colocarse bajo el mismo nodo de Transformada. El campo diTextura especifica la textura con profundidad (TexturaSimple o TexturaDePunto), que se asociará a la región definida en el nodo de ImagenConProfundidad.

El nodo de ImagenOctárbol define una estructura de octárbol y sus texturas proyectadas. El campo ResoluciónOctárbol especifica el máximo número de hojas terminales del octárbol a lo largo de un lado del cubo circundante. El campo octárbol especifica un conjunto de nodos internos del octárbol. Cada nodo interno está representado por un octeto. Un 1 en el i-ésimo bit de este octeto significa que existen los nodos hijos para el i-ésimo hijo de ese nodo interno, mientras que un 0 significa que no existen. El orden de los nodos internos del octárbol será el orden del recorrido por amplitud primero del octárbol. El orden de ocho hijos de un nodo interno se muestra en la FIG. 14 (b). El campo ÍndiceImagenVoxel contiene una formación de índices de imagen asignados al voxel. En la etapa de representación, el color atribuido a una hoja terminal del octárbol se determina proyectando ortográficamente la hoja terminal sobre una de las imágenes con un índice específico. Los índices se almacenan a la manera del octárbol: si puede usarse una imagen específica para todas las hojas terminales contenidas en un voxel específico, el voxel que contiene el índice de la imagen se emite hacia el flujo; en caso contrario, se emite el voxel que contiene un código fijo de 'subdivisión adicional', lo que significa que el índice de imagen se especificará por separado para cada hijo del voxel actual (de la misma forma recursiva). Si el ÍndiceImagenVoxel está vacío, entonces los índices de imagen se determinan durante la etapa de representación. El campo de imágenes especifica un conjunto de nodos de ImagenConProfundidad, con TexturaSimple para el campo diTextura. Sin embargo, no se usan el campo PlanoCercano ni el campo PlanoLejano del nodo ImagenConProfundidad, ni el campo profundidad en el nodo de Textura Simple.

Los procedimientos de representación para los formatos de DIBR no son parte de la AFX, pero es necesario explicar las ideas usadas para lograr la simplicidad, velocidad y calidad de la representación de objetos por DIBR. Nuestros procedimientos de representación se basan en splats, pequeños parches planos de color usados como 'primitivas de representación'. Dos enfoques esbozados más adelante se orientan a dos representaciones distintas: ImagenConProfundidad e ImagenOctárbol. En nuestra implementación, las funciones de OpenGL se emplean para la generación de splats, a fin de acelerar la representación. No obstante, también es posible la representación por software, y permite el cálculo optimizado, usando la estructura sencilla de ImagenConProfundidad o de ImagenOctárbol.

El procedimiento que usamos para representar objetos de ImagenConProfundidad es extremadamente sencillo. Debería mencionarse, sin embargo, que depende de las funciones de OpenGL y que funciona mucho más rápido con la ayuda de un acelerador de hardware. En este procedimiento, transformamos todos los píxeles con profundidad, de TexturasSimples

y TexturasDePunto que haya que representar, en puntos tridimensionales, luego colocamos pequeños polígonos (splats) en estos puntos y aplicamos funciones de representación de OpenGL. El seudocódigo de este procedimiento para el caso de TexturaSimple se da en la FIG. 28. El caso de TexturaDePunto se trata exactamente de la misma manera.

El tamaño del splat debe adaptarse a la distancia entre el punto y el observador. Usamos el siguiente enfoque sencillo. Primero, el cubo circundante del objeto tridimensional dado se subdivide en una rejilla uniforme grosera. El tamaño del splat se calcula para cada celda de la rejilla, y este valor se usa para los puntos dentro de la celda. El cálculo se realiza de la siguiente manera:

- Representar la celda sobre la pantalla por medio de OpenGL.

- Calcular la longitud L de la mayor diagonal de proyección (en píxeles).

- Estimar D (diámetro de splat) como $C\frac{L}{N}$, donde N es un número promedio de puntos por lado de celda, y C es una constante heurística, aproximadamente igual a 1,3.

Quisiéramos subrayar que este procedimiento podría ciertamente mejorar con cálculos más precisos del radio, splats más complejos y generación de antialias. Sin embargo, incluso este enfoque sencillo proporciona una buena calidad visual.

El mismo enfoque funciona para ImagenOctárbol, donde los nodos del octárbol en uno de los niveles más bastos se usan en los cálculos anteriores del tamaño de splat. Sin embargo, para la ImagenOctárbol la información de color debería asociarse primero al conjunto de voxels. Esto puede hacerse muy fácilmente, porque cada voxel tiene su correspondiente índice de imagen de referencia. La posición de píxeles en una imagen de referencia también se conoce durante el análisis sintáctico del flujo del octárbol. En cuando se determinan los colores de los voxels de la ImagenOctárbol, se estiman los tamaños de splat y se usa la representación basada en OpenGL, según lo descrito anteriormente.

Los formatos de DIBR se han implementado y probado en varios modelos tridimensionales. Uno de los modelos (“Torre”) se obtuvo escaneando un objeto físico real (se usó un escáner tridimensional en color de Cyberware), los otros se convirtieron a partir del paquete de demostración 3DS-MAX. Las pruebas se realizaron en un equipo con Pentium-IV de 1,8 GHz, con acelerador de OpenGL.

En las siguientes subsecciones, explicamos los procedimientos de conversión de formatos poligonales a formatos de DIBR, y luego presentamos los resultados de modelización, representación y compresión de los distintos formatos de DIBR. La mayoría de los datos son para modelos de ImagenConProfundidad e ImagenOctárbol; estos formatos tienen versiones animadas y pueden comprimirse efectivamente. Todos los modelos presentados han sido construidos con la cámara ortográfica, ya que, en general, es la manera preferible de representar objetos compactos. Obsérvese que la cámara en perspectiva se usa mayormente para la representación económica de DIBR de los entornos distantes.

La generación del modelo de DIBR comienza con la obtención de un número suficiente de TexturasSimples. Para un objeto poligonal, se calculan las TexturasSimples, mientras que para el objeto del mundo real, los datos se obtienen de cámaras digitales y dispositivos de escaneo. La siguiente etapa depende del formato de DIBR que queramos usar.

La ImagenConProfundidad es simplemente una unión de las TexturasSimples obtenidas. Aunque los mapas de profundidades pueden almacenarse en forma comprimida, sólo es aceptable la compresión sin pérdida, ya que incluso una distorsión pequeña en la geometría es a menudo sumamente notable.

Las imágenes de referencia pueden almacenarse en forma comprimida con pérdida, pero en este caso se requiere un preprocesamiento. Si bien es generalmente tolerable usar procedimientos populares como la compresión con pérdida de JPEG, las distorsiones limítrofes se tornan más notables en las vistas de objetos tridimensionales generadas – especialmente debido a los límites entre el objeto y el fondo de la imagen de referencia, donde el color de fondo parece ‘derramarse’ en el objeto. La solución que hemos usado para afrontar el problema es extender la imagen en los bloques limítrofes hacia el fondo, usando el color promedio del bloque y la decadencia rápida de la intensidad, y aplicar luego la compresión de JPEG. El efecto se parece al ‘estrujamiento’ de la distorsión hacia el fondo, donde es inocuo, ya que los píxeles de fondo no se usan para la representación. Los límites internos en las imágenes de referencia comprimidas con pérdida también pueden producir distorsiones, pero estas son generalmente menos visibles.

Para generar modelos de ImagenOctárbol, usamos una representación intermedia basada en puntos (PBR). El conjunto de puntos que constituyen la PBR es la unión de los puntos coloreados obtenidos por el desplazamiento de píxeles en imágenes de referencia, en distancias especificadas en los correspondientes mapas de profundidades. Las TexturasSimples originales deberían construirse de modo tal que la PBR resultante proporcione una aproximación lo suficientemente precisa de la superficie del objeto. Después de eso, la PBR se convierte en la ImagenOctárbol, según lo esbozado en la FIG. 24, y se usa para generar un nuevo conjunto completo de imágenes de referencia que satisfacen las restricciones impuestas por este formato. A la vez, se genera la estructura adicional de datos ÍndiceImagenVoxel, que

representa índices de imágenes de referencia para voxels de octárboles. En caso de que las imágenes de referencia se almacenaran en formatos con pérdida, se preprocesan primero según lo explicado en la subsección anterior. Además, dado que la estructura del TBVO especifica explícitamente el píxel que contiene su color para cada voxel, los píxeles redundantes se descartan, lo que reduce adicionalmente el volumen de ÍndiceImagenVoxel. Ejemplos de las imágenes de referencia, originales y procesadas en el formato JPEG, se muestran en la FIG. 29.

Obsérvese que la degradación de calidad debida a la compresión con pérdida es despreciable para ImágenesOctárbol, pero a veces aún es notable para los objetos de ImagenConProfundidad.

Los modelos de TexturaDePunto se construyen usando la proyección del objeto sobre un plano de referencia, según se explica en la Sección 2.1. Si esto no produce bastantes muestras (lo que puede ser el caso para las partes superficiales casi tangentes al vector de proyección), se construyen TexturasSimples adicionales para proporcionar más muestras. El conjunto obtenido de muestras se reorganiza luego en la estructura TexturaDePunto.

Se presentarán ahora los datos sobre la velocidad de representación.

La velocidad de representación de la ImagenConProfundidad “Palmera512” es de alrededor de 2 tramas por segundo (obsérvese que son 21 Texturas Simples), mientras que otros modelos estáticos que probamos con un tamaño de imagen de referencia de 512 se representan con entre 5 y 6 tramas por segundo. Obsérvese que la velocidad de representación depende mayormente del número y la resolución de las imágenes de referencia, pero no de la complejidad de la escena. Esta es una ventaja importante sobre las representaciones poligonales, especialmente en el caso animado. La ImagenOctárbol animada “Dragón512” se visualiza a razón de 24 tramas por segundo (fps).

El modelo de ImagenConProfundidad “Ángel256” se muestra en la FIG. 22. Las FIGs. 30 a 34 muestran otros diversos modelos de DIBR y poligonales. La FIG. 30 compara la apariencia del modelo “Morton”, poligonal y de ImagenConProfundidad. El modelo de ImagenConProfundidad usa imágenes de referencia en el formato JPEG, y la representación se realiza por la generación más sencilla de splats, descrita en la Sección 5, pero la calidad de imagen es bastante aceptable. La FIG. 31 compara dos versiones del modelo de “Torre” escaneado. Los puntos negros en la parte superior del modelo se deben a datos de entrada ruidosos. La FIG. 32 demuestra un modelo de “Palmera” más complejo, compuesto por 21 Texturas Simples. También muestra buena calidad, aunque las hojas terminales son, en general, más anchas en el original 3DS-MAX, lo que es una consecuencia de la generación simplificada de splats.

La FIG. 33 presenta una trama tridimensional de la animación de la ImagenOctárbol “Dragón512”. La FIG. 34 demuestra la capacidad de un formato de TexturaDePunto para proporcionar modelos de excelente calidad.

La estructura de nodo basada en imagen con profundidad según la presente invención incluye el nodo de TexturaSimple, el nodo de TexturaDePunto, el nodo de ImagenConProfundidad y el nodo de ImagenOctárbol. El nodo de ImagenConProfundidad está compuesto por información de profundidad y una imagen en color. La imagen en color se selecciona entre el nodo de TexturaSimple y el nodo de TexturaDePunto.

Cuando un objeto se visualiza desde seis puntos de vista (frente, fondo, plano, retaguardia, lados izquierdo y derecho), el objeto puede representarse con seis pares de nodos de TexturaSimple. La especificación del nodo de TexturaSimple se muestra en la FIG. 26.

Con referencia a la FIG. 26, el nodo de TexturaSimple se compone de un campo de Textura en el cual se registra una imagen en color que contiene el color para cada píxel, y un campo de profundidad en el cual se registra la profundidad para cada píxel. El nodo de TexturaSimple define una única textura de IBR. Aquí, una textura significa una imagen plana coloreada.

Una imagen plana, que contiene el color para cada píxel que forma la imagen, está en el campo de textura. La profundidad para cada píxel que forma la imagen se registra en el campo de profundidad. Un conjunto de profundidades en el campo de profundidad forma las imágenes con profundidad correspondientes a la imagen plana en el campo de textura. Las imágenes con profundidad son imágenes planas representadas en escalas de grises, según las profundidades. En el caso de un formato de vídeo para generar objetos animados, la información de profundidad y la información de color son secuencias múltiples de tramas de imagen.

La imagen plana en el campo de textura (es decir, la imagen coloreada) y la imagen plana en el campo de profundidad (es decir, la imagen representada en escalas de grises) constituyen un nodo de TexturaSimple. La FIG. 20 muestra objetos “Morton” representados por los nodos de TexturaSimple para puntos de vista frontales. En conclusión, los objetos están representados por seis nodos de TexturaSimple, que son pares de imágenes generadas para seis puntos de vista. La FIG. 22 muestra objetos del “Ángel” representados por seis nodos de TexturaSimple.

La imagen en color puede representarse con nodos de TexturaDePunto. La FIG. 21 muestra texturas de Punto generadas por la proyección de un objeto sobre un plano de referencia (en este caso, un plano separado por una distancia predeterminada del objeto, para enfrentar la cara trasera del objeto).

La FIG. 26 también muestra la especificación de un nodo de TexturaDePunto. Con referencia a la FIG. 26, un nodo de TexturaDePunto se compone de un campo de tamaño, un campo de resolución, un campo de profundidad y un campo de color. La información de tamaño de un plano de imagen se registra en el campo de tamaño. El campo de tamaño se compone de campos de ancho y altura, donde se registran, respectivamente, el ancho y la altura del plano de imagen. El tamaño del plano de imagen se fija en un tamaño suficiente como para cubrir el objeto entero proyectado sobre el plano de referencia.

La información de resolución sobre la profundidad para cada píxel se registra en el campo de resolución. Por ejemplo, cuando se registra un número “8” en el campo de resolución, la profundidad de un objeto se representa con 256 escalas, en base a la distancia desde el plano de referencia.

Múltiples elementos de información de profundidad sobre cada píxel se registran en el campo de profundidad. La información de profundidad es una secuencia de números de píxeles proyectados sobre el plano de imagen y las profundidades para los respectivos píxeles. La información de color sobre cada píxel se registra en el campo de color. La información de color es una secuencia de colores correspondiente a los respectivos píxeles proyectados sobre el plano de la imagen.

La información de punto de vista que constituye el nodo de ImagenConProfundidad incluye varios campos tales como el punto de vista, la visibilidad, la proyección, el procedimiento o la distancia.

En el campo de punto de vista, se registran los puntos de vista desde los cuales se visualiza un plano de imagen. El campo de punto de vista tiene campos de posición y orientación, donde se registran la posición y orientación del punto de vista. La posición en el campo de posición es una ubicación relativa del punto de vista con respecto al origen (0, 0, 0) del sistema de coordenadas, mientras que la orientación en el campo de orientación es una magnitud de rotación del punto de vista con respecto a la orientación por omisión.

En el campo de visibilidad, se registra un área de visibilidad desde el punto de vista hasta el plano de imagen. En el campo de procedimiento de proyección, se registra un procedimiento de proyección desde el punto de vista hasta el plano de imagen. En la presente invención, el procedimiento de proyección incluye un procedimiento de proyección ortogonal en el cual el área de visibilidad está representada por el ancho y la altura, y un procedimiento de proyección en perspectiva, en el cual el área de visibilidad está representada por un ángulo horizontal y un ángulo vertical. Cuando se selecciona el procedimiento de proyección ortogonal, es decir, cuando el campo del procedimiento de proyección se fija en TRUE, el ancho y la altura del área de visibilidad corresponden, respectivamente, al ancho y a la altura de un plano de imagen. Cuando se selecciona el procedimiento de proyección en perspectiva, los ángulos horizontal y vertical del área de visibilidad corresponden a los ángulos formados con los lados horizontal y vertical por las vistas que varían desde un punto de vista hasta el plano de imagen.

En el campo de distancia, se registran una distancia desde un punto de vista hasta un plano limítrofe más cercano y una distancia desde el punto de vista hasta un plano limítrofe más lejano. El campo de distancia se compone de un campo de PlanoCercano y un campo de PlanoLejano. El campo de distancia define un área para la información de profundidad.

Las FIGs. 35A y 35B son diagramas que muestran las relaciones de los respectivos nodos al representar un objeto en un formato de ImagenConProfundidad, con nodos de TexturaSimple y nodos de TexturaDePunto, respectivamente.

Con referencia a la FIG. 35A, el objeto puede representarse con conjuntos de nodos de ImagenConProfundidad, correspondientes a seis puntos de vista. Cada uno de los respectivos nodos de ImagenConProfundidad consiste en información de punto de vista y TexturaSimple. La TexturaSimple consiste en un par de imagen en color e imagen con profundidad.

Con referencia a la FIG. 35B, el objeto puede representarse con un nodo de ImagenConProfundidad. La especificación del nodo de ImagenConProfundidad se describe como antes. Un nodo de TexturaDePunto se compone de información de plano, con información sobre un plano sobre el cual se proyecta el objeto, e información de profundidad e información de color de diversos puntos de los objetos proyectados sobre el plano de imagen.

En un nodo de ImagenOctárbol, un objeto se representa con la estructura de nodos internos que constituyen los voxels que contienen el objeto y las imágenes de referencia. La especificación del nodo de ImagenOctárbol se muestra en la FIG. 26.

Con referencia a la FIG. 26, el nodo de ImagenOctárbol incluye campos de ResoluciónOctárbol, octárbol, ÍndiceImagenVoxel e imágenes.

En el campo de ResoluciónOctárbol, se registra el número máximo de hojas terminales de octárbol a lo largo de un lado del cubo circundante que contiene el objeto. En el campo de octárbol, se registra una estructura de nodo interno. Un nodo interno es un nodo para un subcubo generado después de subdividir el cubo circundante que contiene el objeto entero. La subdivisión de cada subcubo se lleva a cabo iterativamente para generar 8 subcubos, hasta que se alcanza un número

predeterminado de subcubos. En el caso de efectuar iterativamente la subdivisión 3 veces, suponiendo que un nodo para un subcubo después de la segunda iteración de subdivisión se menciona como el nodo actual, un nodo para un subcubo después de la primera iteración de subdivisión y un nodo para un subcubo después de la tercera subdivisión se mencionan como un nodo padre y un nodo hijo, respectivamente. El orden de 8 subcubos divididos está dado por el orden de prioridad en el ancho. La FIG. 14 muestra un procedimiento de asignación de números de prioridad de subcubos. Cada nodo interno está representado por un octeto. La información de nodos registrada en los flujos de bits que constituyen el octeto representa la presencia o ausencia de nodos hijos de nodos hijos pertenecientes al nodo interno.

En el campo de índice, se registran los índices de imágenes de referencia correspondientes a los respectivos nodos internos. En el campo de imagen, se registran las imágenes de referencia correspondientes a los índices registrados en el campo de índice. Las imágenes de referencia son nodos de ImagenConProfundidad, y la estructura de las mismas se describe como antes.

La FIG. 36 es un diagrama que muestra la estructura de un nodo pertinente de ImagenOctárbol al representar un objeto usando nodos de ImagenOctárbol.

Con referencia a la FIG. 36, los nodos de ImagenOctárbol están encapsulados por envoltorios de bits. Cada envoltorio de bits incluye un nodo de ImagenOctárbol. Cuando un objeto se representa en nodos de TexturaSimple, el nodo de ImagenOctárbol incluye 6 nodos de ImagenConProfundidad, conteniendo cada nodo de ImagenConProfundidad un nodo de TexturaSimple. Por otra parte, cuando un objeto se representa con nodos de TexturaDePunto, el nodo de ImagenOctárbol incluye un único nodo de ImagenConProfundidad.

La presente invención puede implementarse en un medio de registro legible por ordenador, por medio de códigos legibles por ordenador. El medio de registro legible por ordenador incluye toda clase de aparatos de registro de los cuales puedan leerse datos legibles por un sistema de ordenador, y los ejemplos del mismo son las memorias ROM, RAM, CD-ROM, las cintas magnéticas, los discos flexibles, los dispositivos de almacenamiento de datos ópticos o similares, y también los datos realizados en una onda portadora, p. ej., desde Internet u otro medio de transmisión. Además, el medio de registro legible por ordenador se distribuye en un sistema de ordenador conectado con una red, a fin de que los códigos legibles por ordenador se almacenen e implementen por medio de un procedimiento distribuido.

Según la presente invención, en las representaciones basadas en imágenes, dado que la información perfecta en un objeto tridimensional coloreado se codifica por medio de un conjunto de imágenes bidimensionales – una estructura sencilla y regular instantáneamente adoptada en procedimientos bien conocidos para el procesamiento y la compresión de imágenes, el algoritmo es simple y puede disponer de soporte por parte del hardware en muchos aspectos. Además, el tiempo de representación para los modelos basados en imágenes es proporcional al número de píxeles en las imágenes de referencia y de salida, pero, en general, no a la complejidad geométrica, como en el caso poligonal. Además, cuando la representación basada en imágenes se aplica a los objetos y escenas del mundo real, la representación de realismo fotográfico de una escena natural se hace posible sin el uso de millones de polígonos y de cálculos caros.

La descripción precedente de una implementación de la invención se ha presentado con fines de ilustración y descripción. No es exhaustiva y no limita la invención a la forma exacta revelada. Son posibles modificaciones y variaciones a la luz de las revelaciones anteriores, o bien pueden adquirirse a partir de la práctica de la invención. El alcance de la invención está definido por las reivindicaciones y sus equivalentes.

REIVINDICACIONES

1. Una estructura de nodos para representar un objeto tridimensional, comprendiendo la estructura de nodos:

un campo de resoluciónoctárbol, en el cual se registra el máximo valor de las hojas terminales del octárbol a lo largo del lado de un cubo circundante que contiene un objeto;

un campo de octárbol en el cual se registra una estructura del nodo interno del octárbol;

un campo de índice de cámara en el cual se registran los índices de voxels del octárbol de una imagen de referencia correspondiente al nodo interno; y

un campo de imagen en el cual se registra la imagen de referencia.

2. La estructura de nodos según la reivindicación 1, en la cual el nodo interno está representado por un octeto, y la información de nodos registrada en los flujos de bits que constituyen el octeto representa la presencia o ausencia de nodos hijos de nodos hijos pertenecientes al nodo interno.

3. La estructura de nodos según la reivindicación 1, en la cual la imagen de referencia es una imagen con profundidad que incluye información de punto de vista y una imagen en color correspondiente a la información de punto de vista.

4. La estructura de nodos según la reivindicación 3, en el cual la información de punto de vista comprende:

un campo de punto de vista en el cual se registran los puntos de vista desde los cuales se visualiza un plano de imagen;

un campo de campoDeVisión en el cual se registra un área de visibilidad desde el punto de vista hasta el plano de imagen; y

un campo ortográfico en el cual se registra un procedimiento de proyección desde el punto de vista hasta el plano de imagen.

5. La estructura de nodos según la reivindicación 4, en la cual el campo de punto de vista comprende:

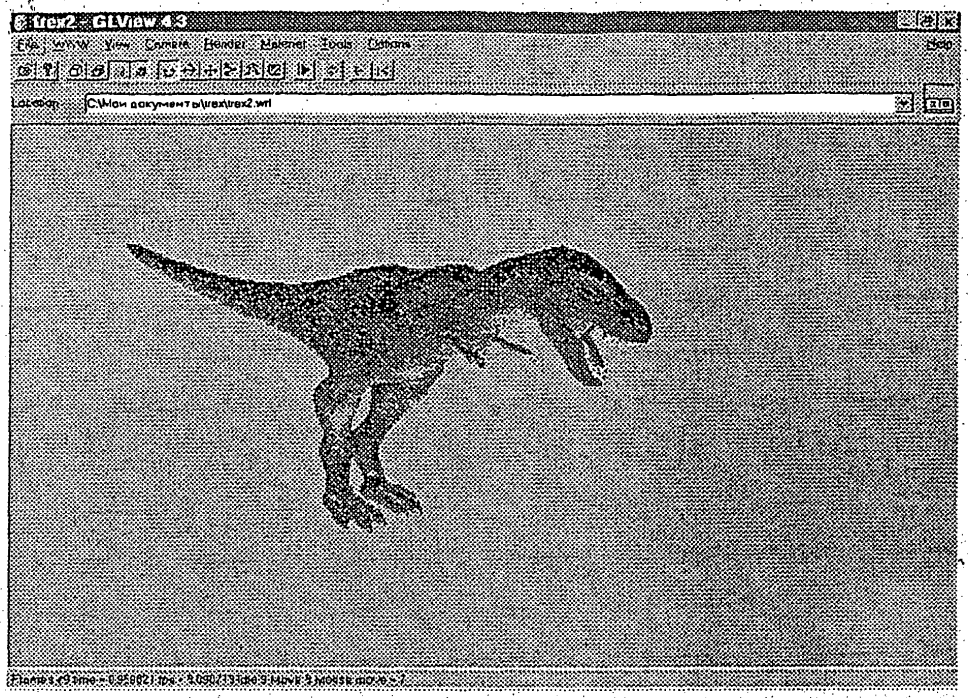
un campo de posición donde se registra la posición de un punto de vista; y

un campo de orientación donde se registra la orientación del punto de vista, siendo la posición una ubicación relativa al origen del sistema de coordenadas, y siendo la orientación una magnitud de rotación con respecto a la orientación por omisión.

6. La estructura de nodos según la reivindicación 4, en la cual el procedimiento de proyección es un procedimiento de proyección ortogonal, y el ancho y la altura del área de visibilidad corresponden, respectivamente, al ancho y a la altura de un plano de imagen.

7. La estructura de nodos según la reivindicación 3, en la cual la imagen en color es una TexturaSimple que consiste en una imagen plana que contiene el color para cada píxel.

FIG. 1


trex2 - GLView 4.3
File www View Camera Render Material Tools Options Help
Location C:\Мои документы\trex\trex2.wrl

(a)

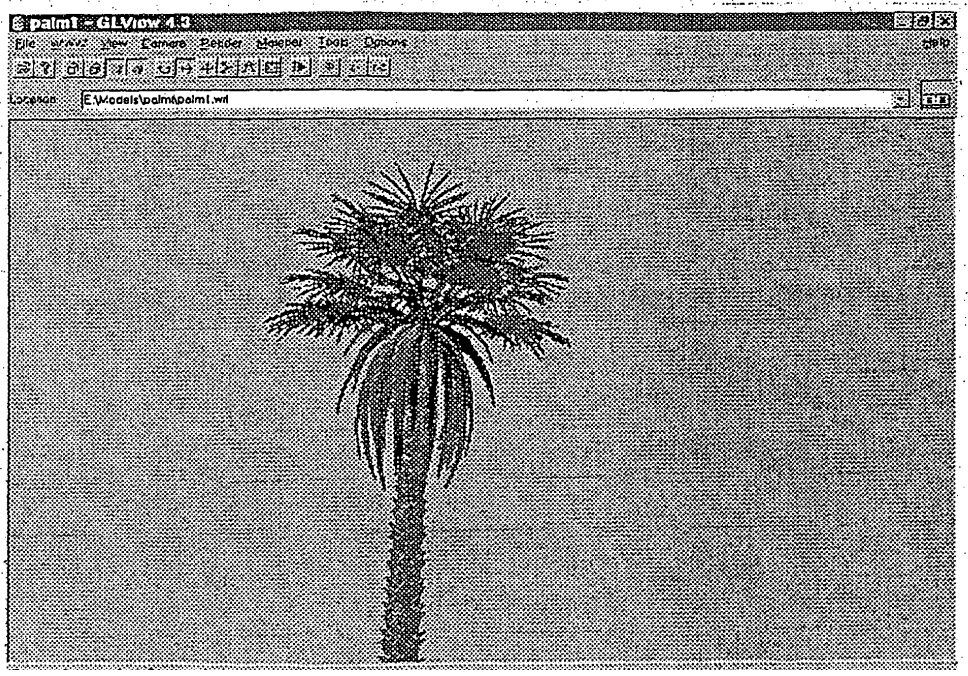

palm1 - GLView 4.3
File www View Camera Render Material Tools Options Help
Location E:\models\palm\palm1.wrl

FIG. 2

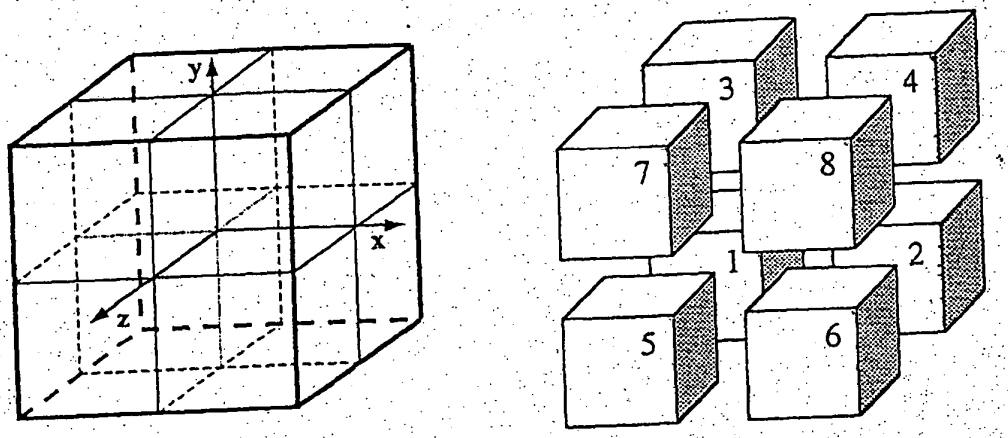

y
x
z
3
4
7
8
1
2
5
6

FIG. 3

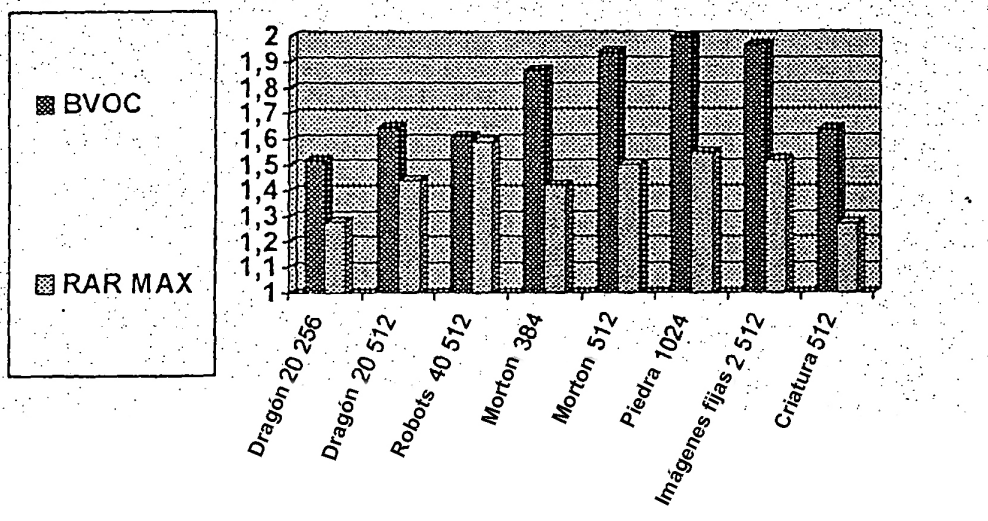

BVOC
RAR MAX
2
1,9
1,8
1,7
1,6
1,5
1,4
1,3
1,2
1,1
1
Dragón 20 256
Dragón 20 512
Robots 40 512
Morton 384
Morton 512
Piedra 1024
Imágenes fijas 2 512
Criatura 512

FIG. 4

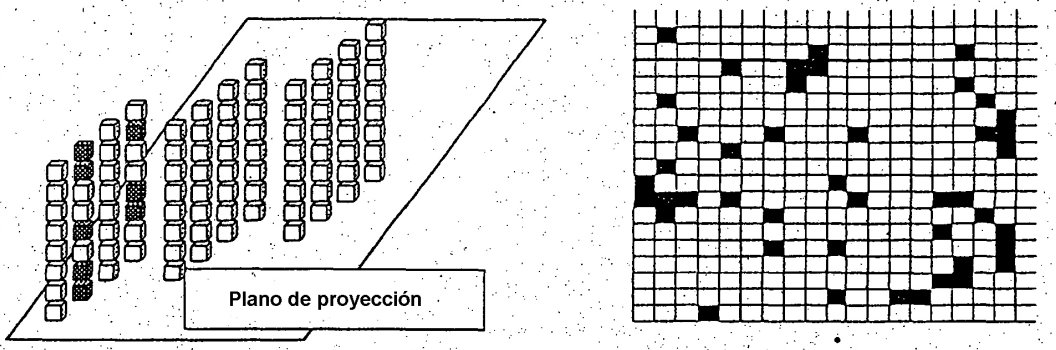
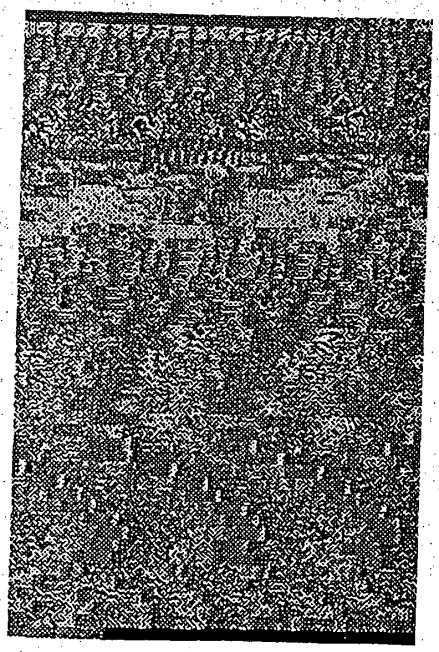


FIG. 5



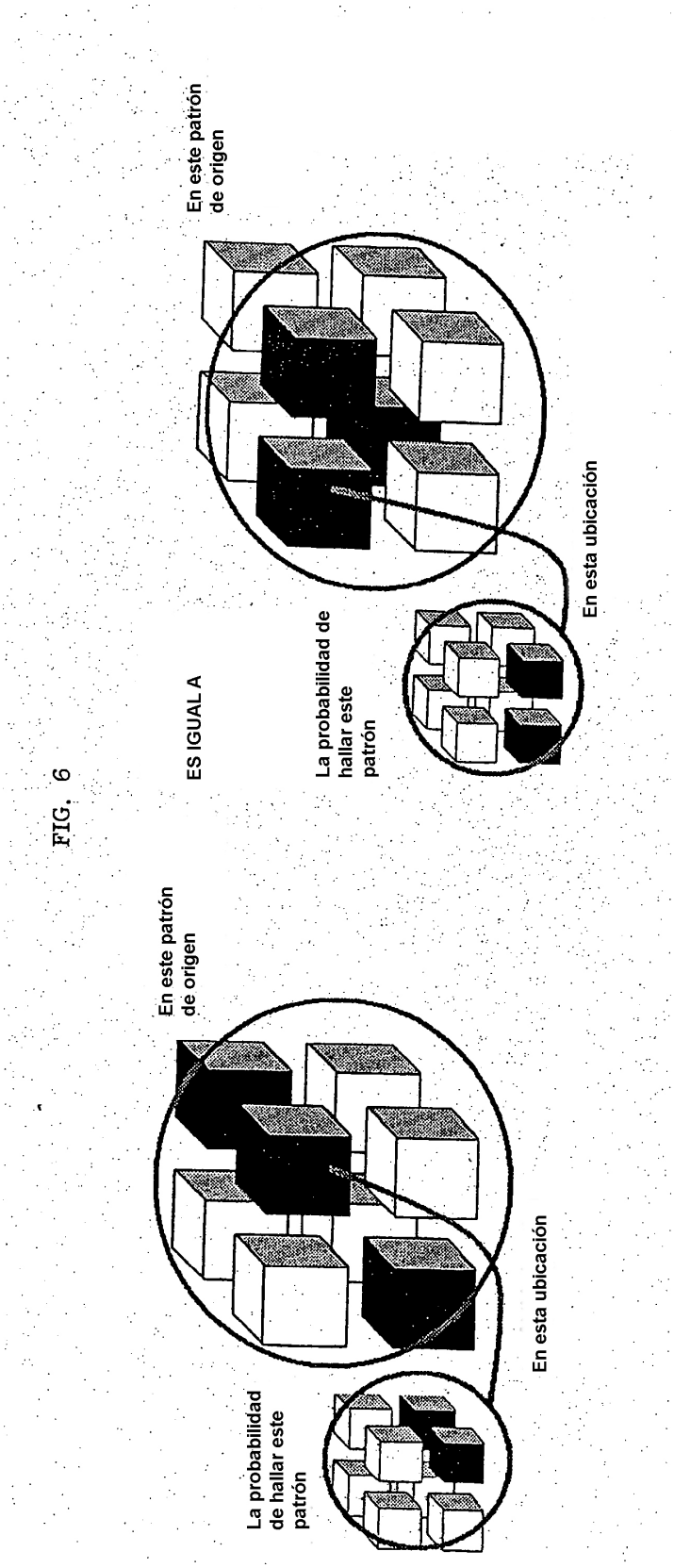
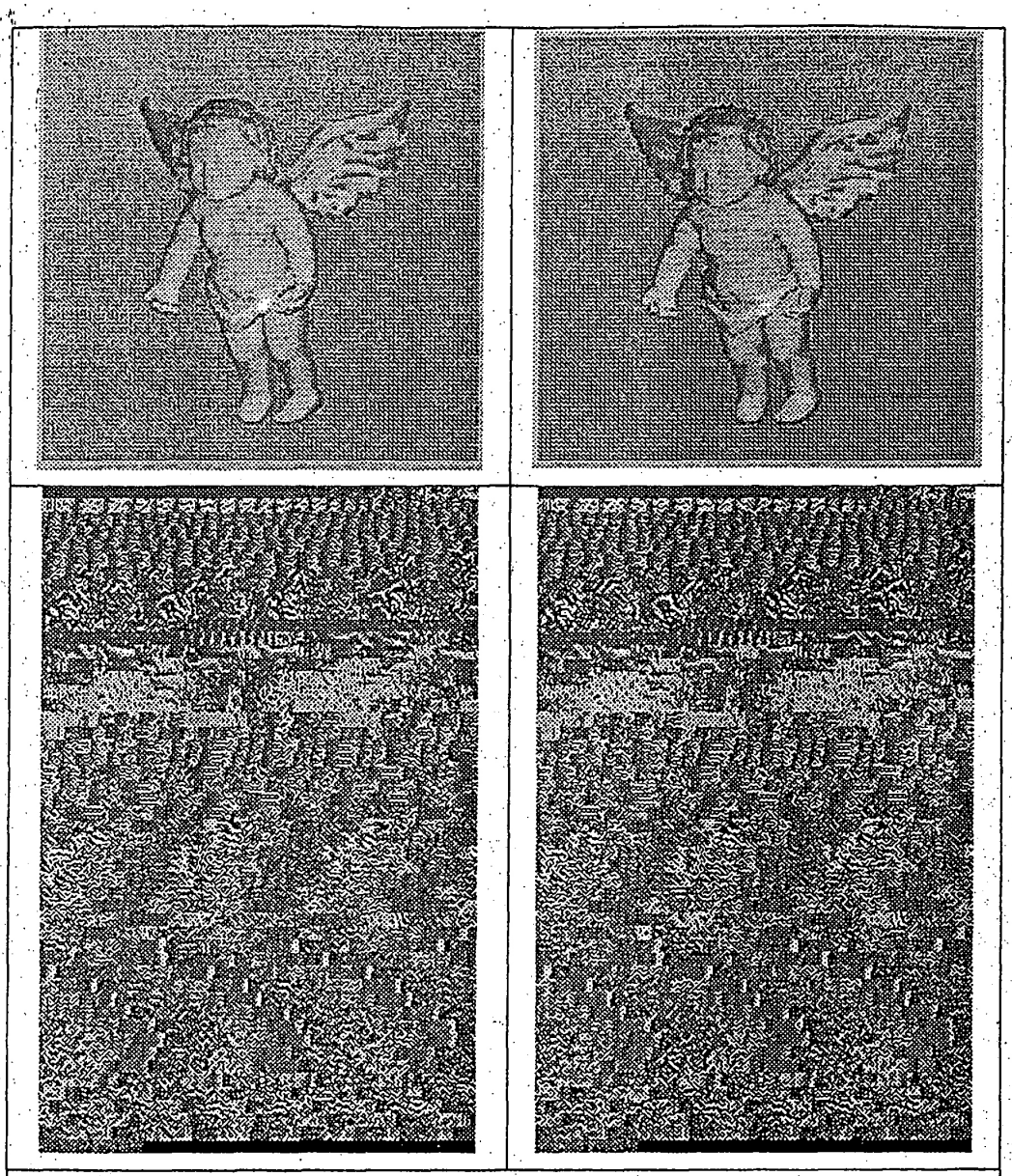

En este patrón de origen
ES IGUAL A
La probabilidad de hallar este patrón
En esta ubicación
En este patrón de origen
La probabilidad de hallar este patrón
En esta ubicación

FIG. 6



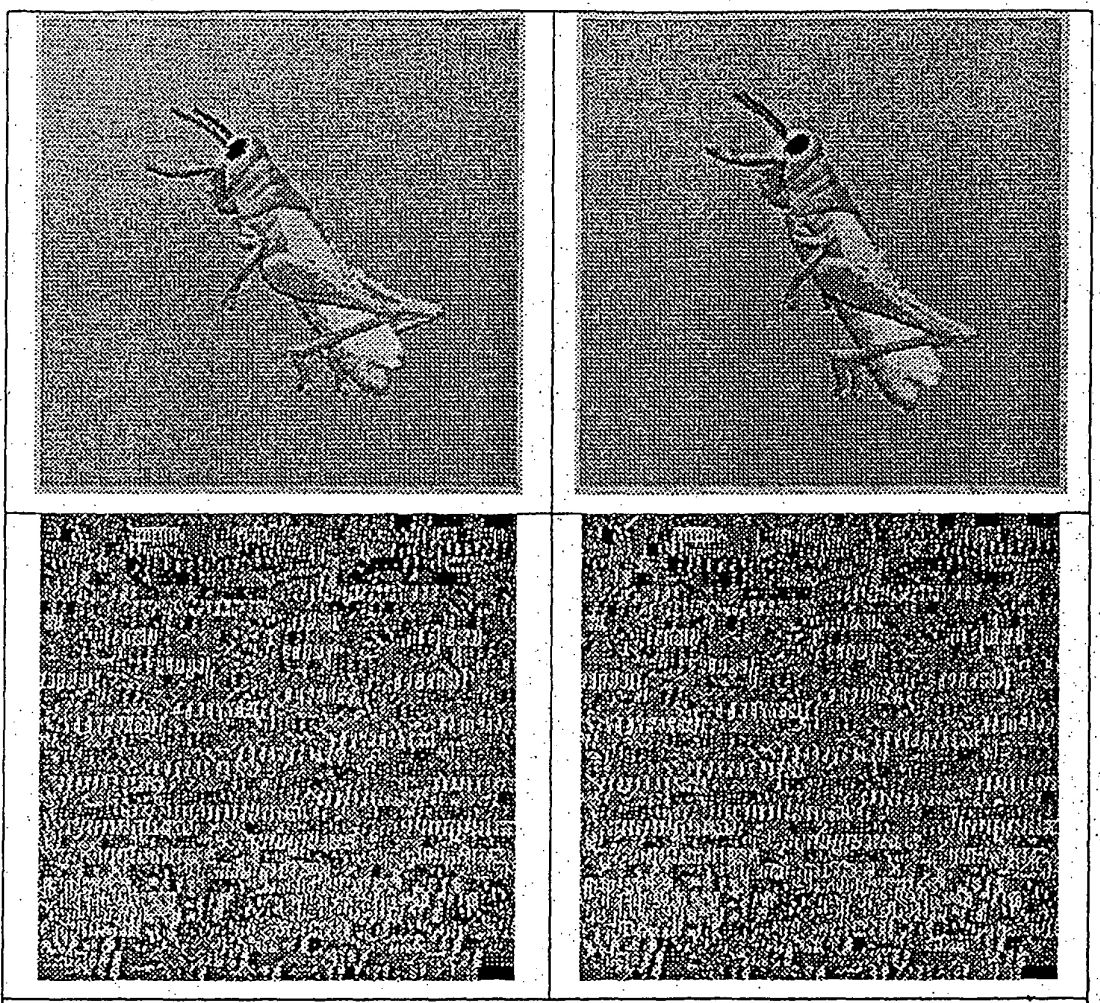
Ángel

Arriba: modelo original (izquierda) y modelo reconstruido (derecha)

Abajo: imagen escaneada original (izquierda), imagen escaneada comprimida (derecha)

Tamaño de fichero LDI: 468.880 octetos
Tamaño del fichero comprimido: 65.876 octetos (geometría de 27.468 octetos + color de 38.408 octetos)
Tasa de compresión: 7,1

FIG. 8



Saltamontes

Arriba: modelo original (izquierda) y modelo reconstruido (derecha)

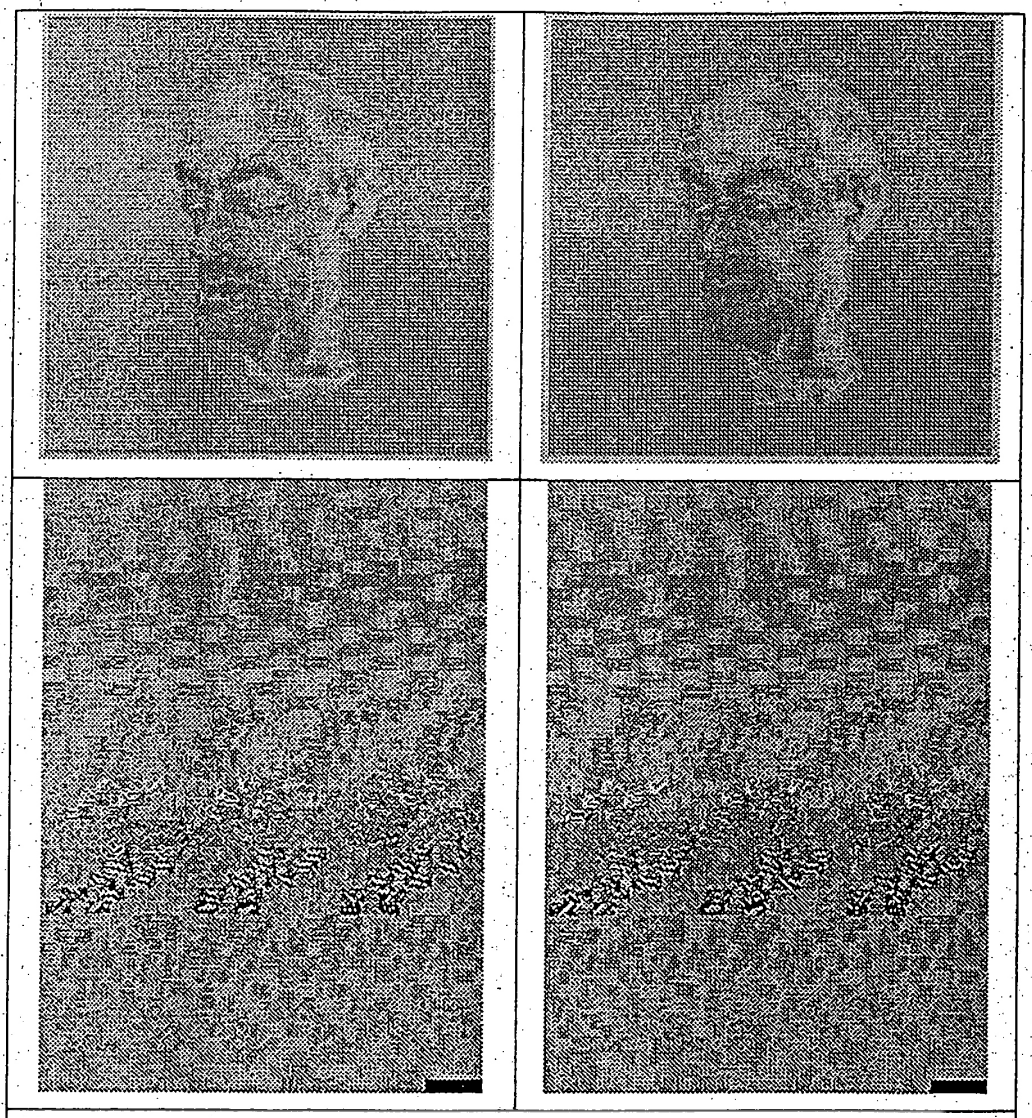
Abajo: imagen escaneada original (izquierda), imagen escaneada comprimida (derecha)

Tamaño de fichero LDI: 327.044 octetos

Tamaño del fichero comprimido: 41.656 octetos (geometría de 17.072 octetos + color de 24.584 octetos)

Tasa de compresión: 7,8

FIG. 9



Morton

Arriba: modelo original (izquierda) y modelo reconstruido (derecha)

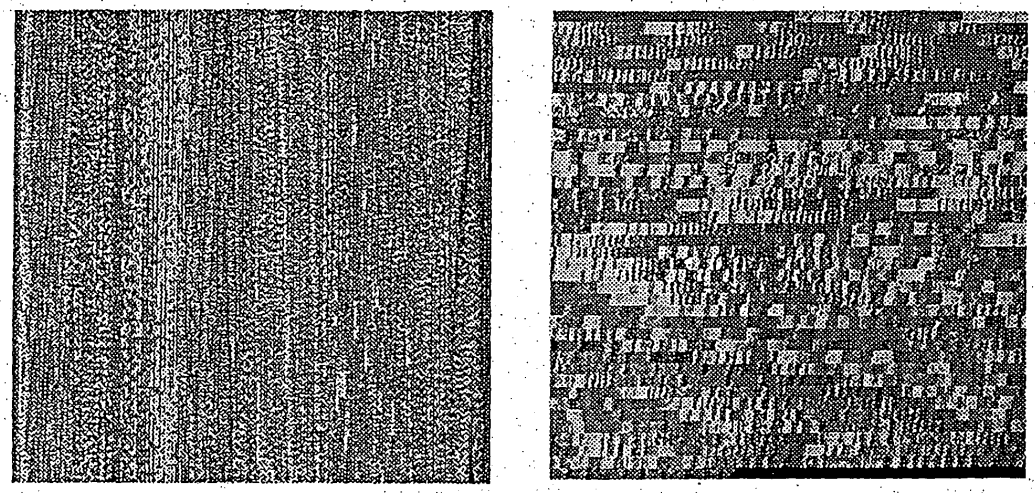
Abajo: imagen escaneada original (izquierda), imagen escaneada comprimida (derecha)

Tamaño de fichero LDI: 433.112 octetos

Tamaño del fichero comprimido: 57.881 octetos (geometría de 23.313 octetos + color de 34.568 octetos)

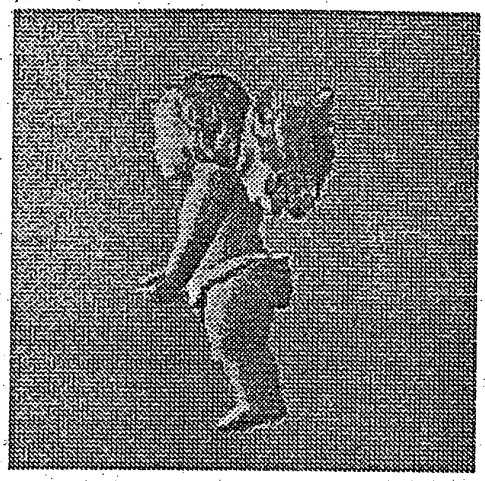
Tasa de compresión: 7,5

FIG. 10

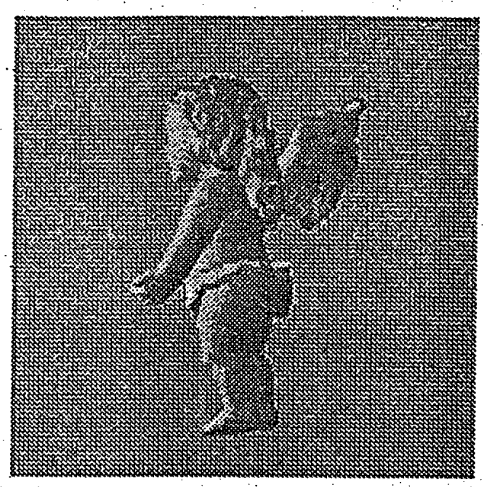


(a) (b)

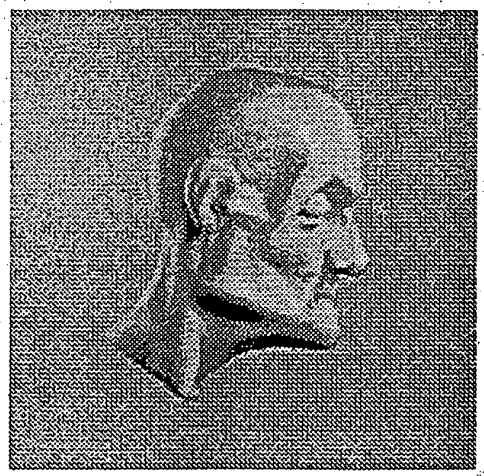
FIG. 11



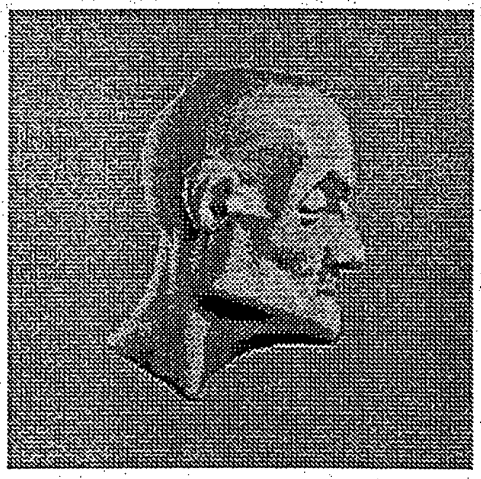
(a)



(b)



(c)



(d)

FIG. 12

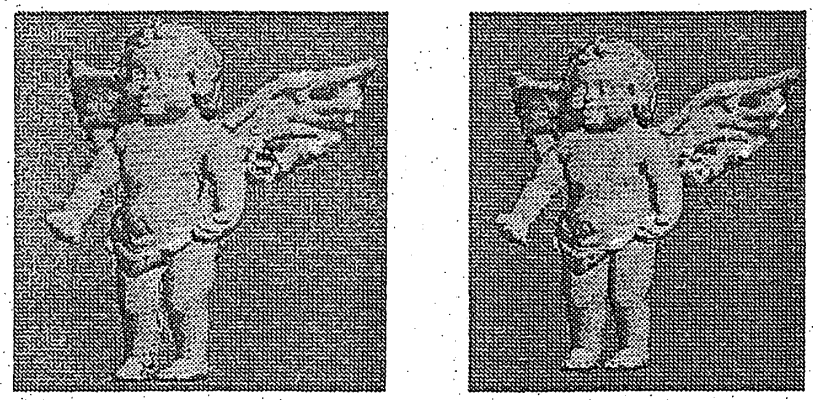


FIG. 13

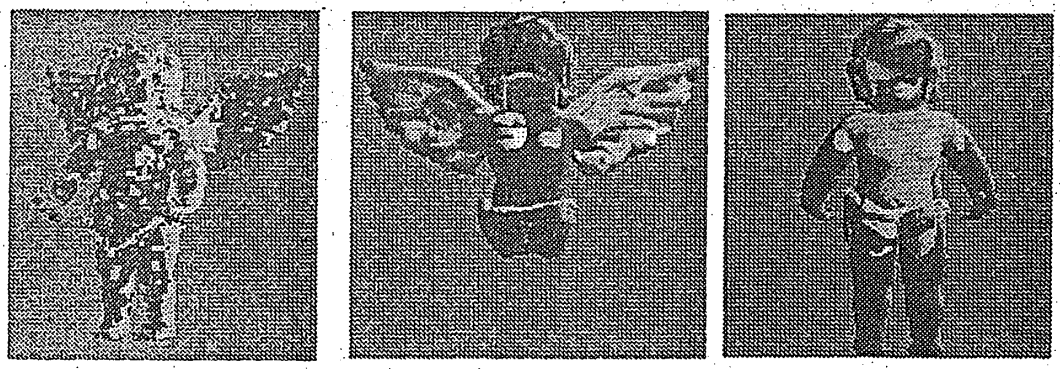


FIG. 14

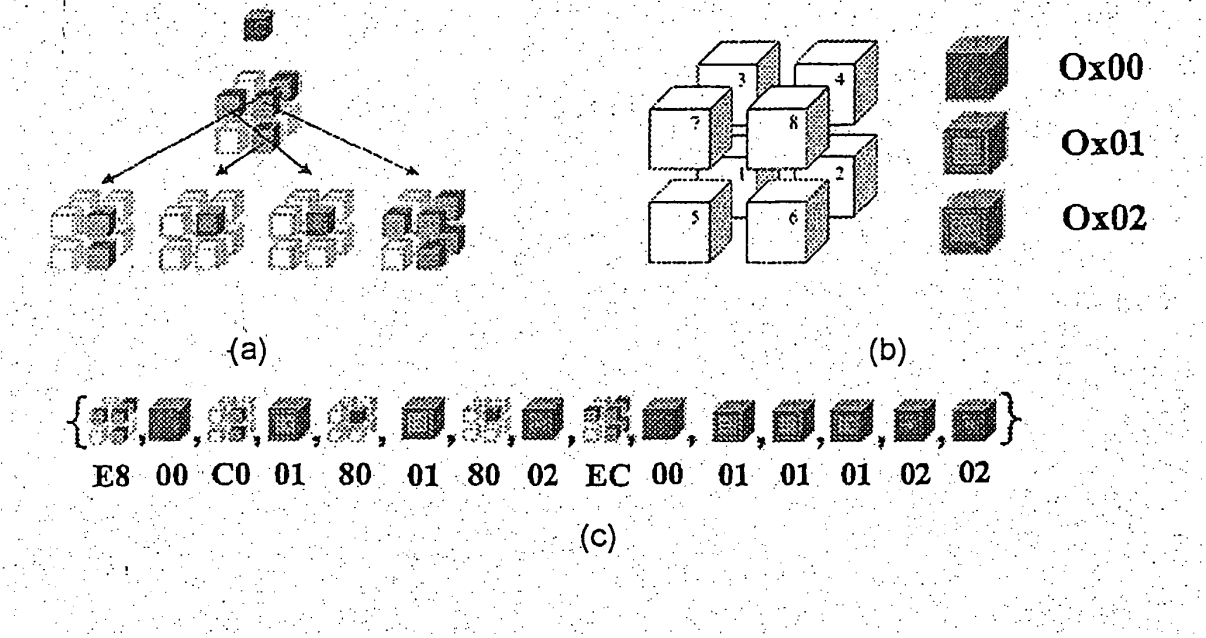


FIG. 15

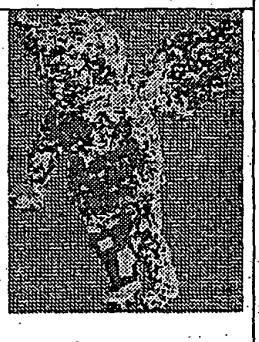
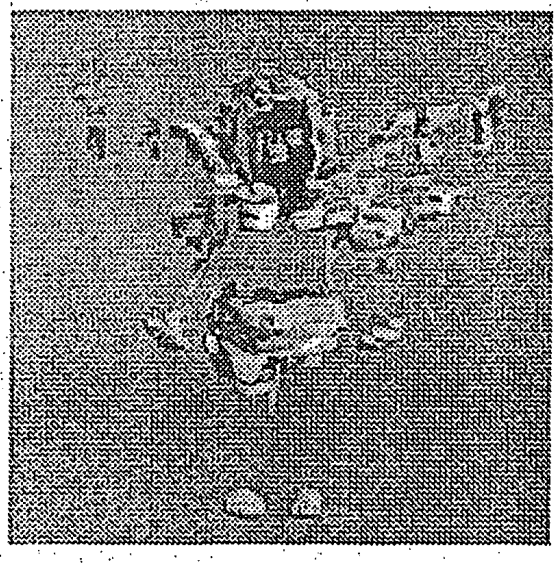
PROCEDIMIENTO	PSNR (DB)	TAMAÑO (TOTAL Y SÓLO GEOMETRÍA)	VISTA	CÁMARAS
BVO	22,7	58K (32K)		
TBVO-6	29,3	75K (38K)		
TBVO-12	33,53	103K (42K)		
TBVO-(6+6)	31,28	106K (42K)		

FIG. 16



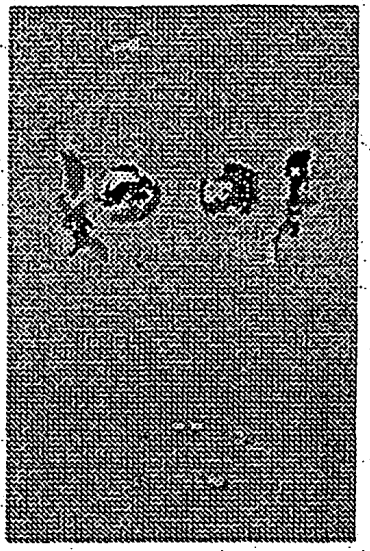


FIG. 17

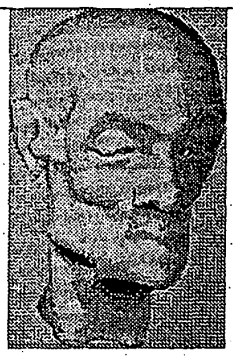
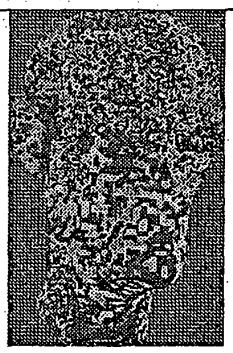
BVO	27,5	146(87K)		
TBVO-6	36,7	192K (103K)		
TBVO-12	39,9	267K (114K)		
TBVO-(6+6)	37,7	171K (88K)		

FIG. 18

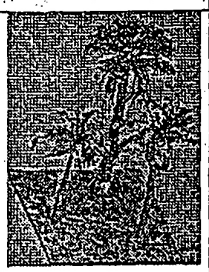
PROCEDIMIENTO	PSNR (DB)	TAMAÑO (TOTAL Y SÓLO GEOMETRÍA)	VISTA	CÁMARAS
BVO	27,0	191(81K)		
TBVO-6	30,9	192K (86K)		
TBVO-6 CON DISTINTAS UBICACIONES DE CÁMARA	32,8	239K (104K)		
TBVO-12	33,30	212K (83K)		
TBVO-6+6	33,34	216K (83K)		

FIG. 19

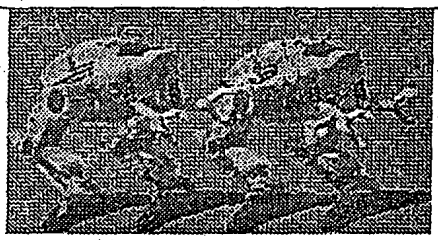
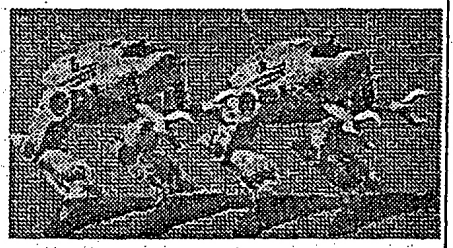
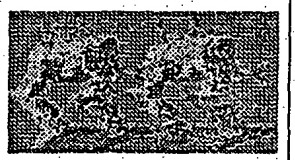
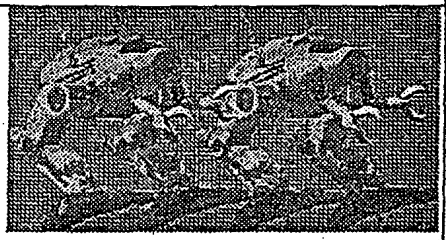
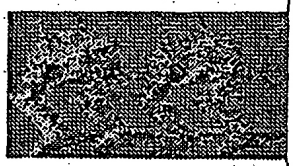
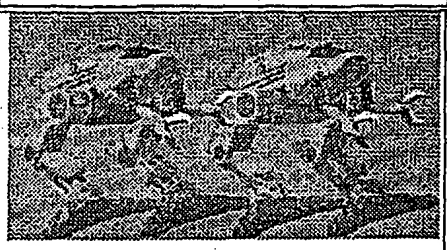
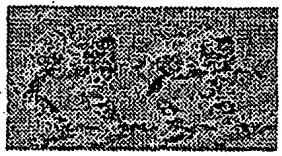
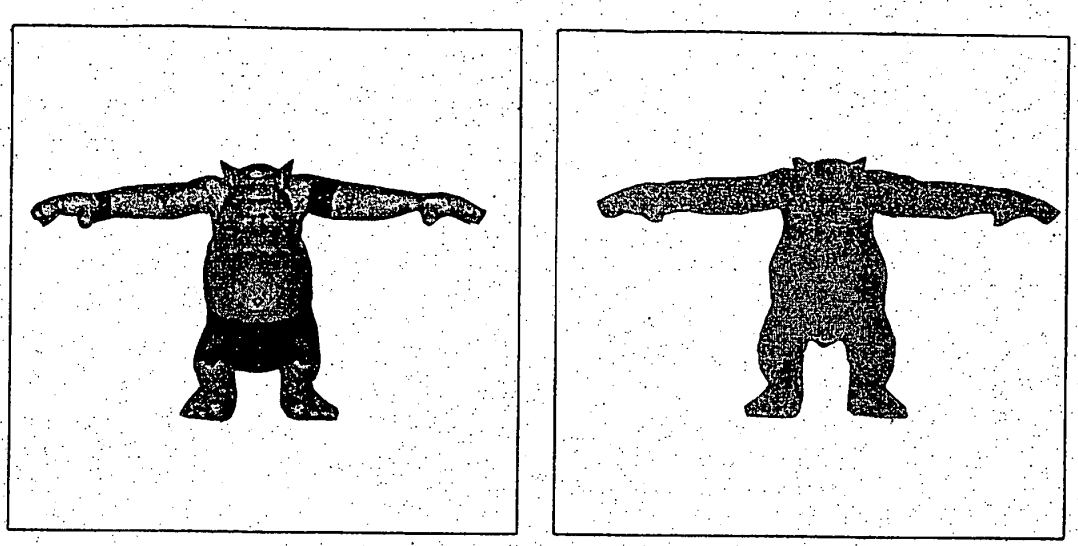
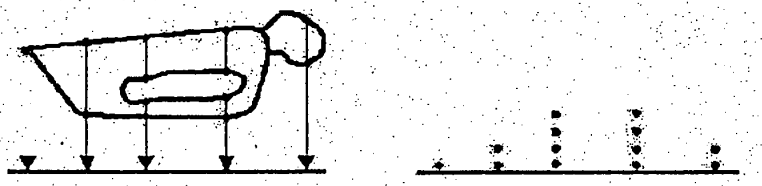
PROCEDIMIENTO	PSNR (DB)	TAMAÑO (TOTAL Y SÓLO GEOMETRÍA)	VISTA	CÁMARAS
BVO	24,4	162(82K)		
TBVO-6	30,9	235K (112K)		
TBVO-12	32,1	244K (121K)		
TBVO-(6+6)	32,55	274K (126K)		

FIG. 20



(a) (b)

FIG. 21



(a) (b)

FIG. 22

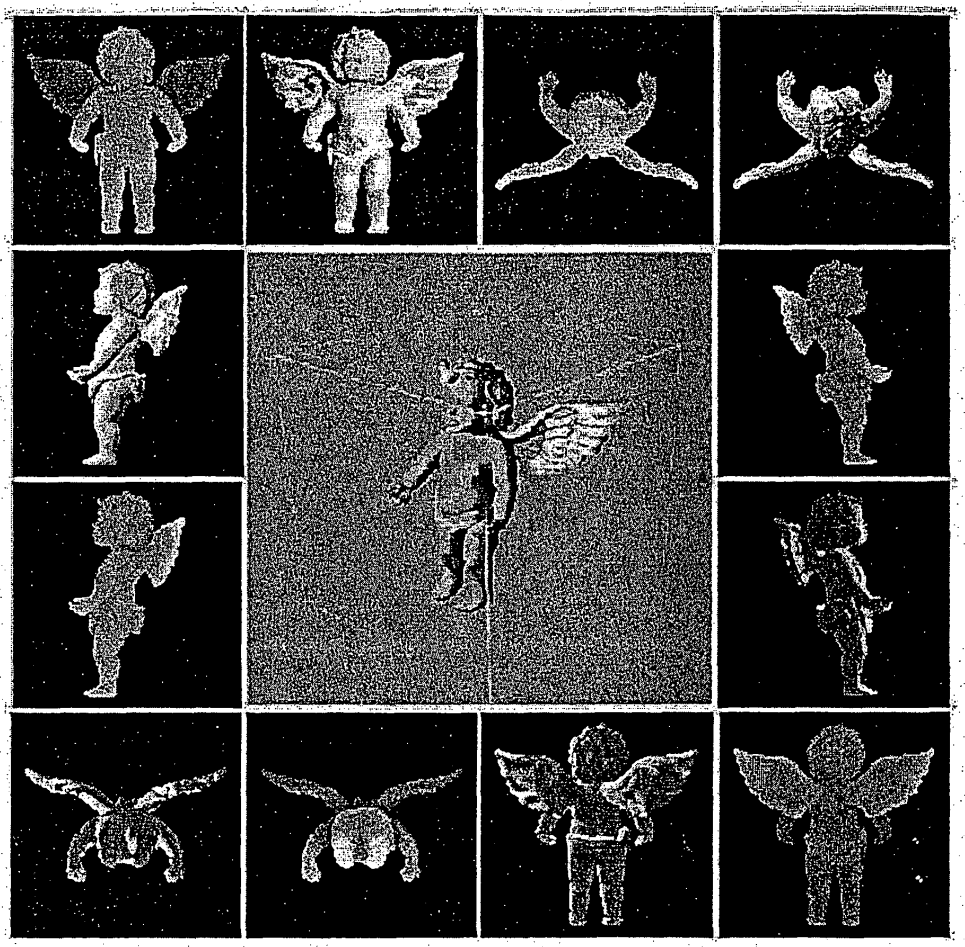
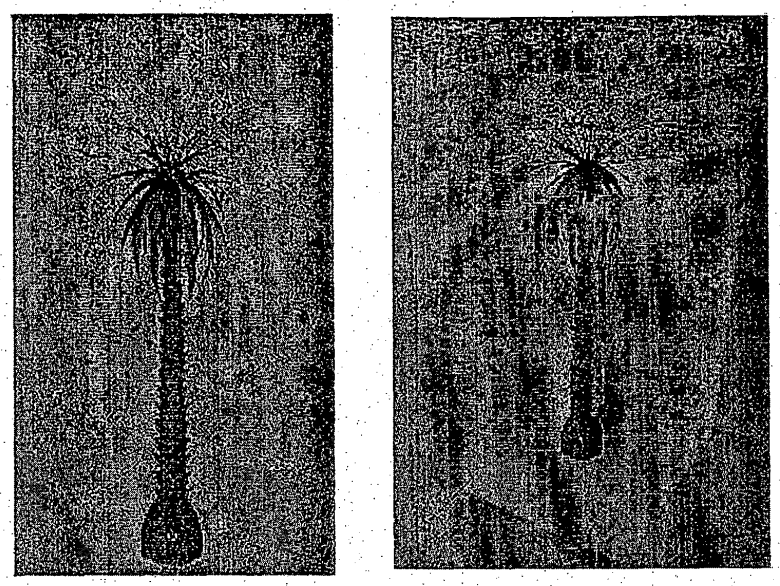
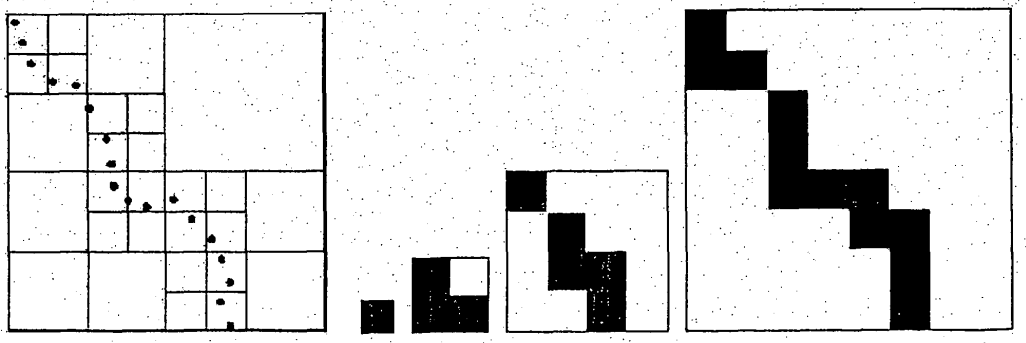


FIG. 23



(a) (b)

FIG. 24



(a) (b)

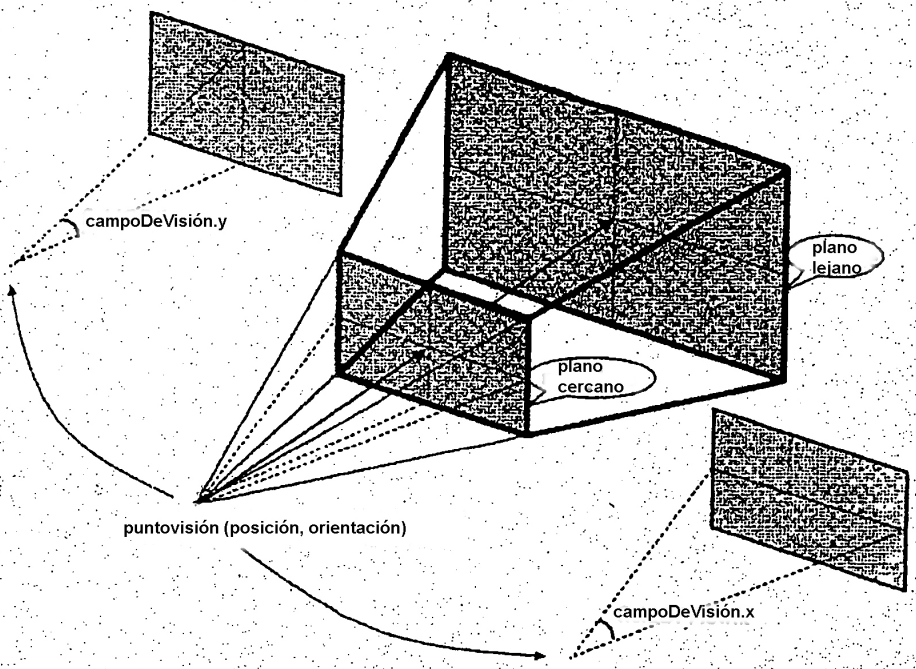
FIG. 25

```
if NodoAct no es nodo terminal
  escribir símbolo-BVO actual correspondiente a este nodo
si todos los hijos tienen idéntico índice de imagen (símbolo de textura)
  si el padre de NodoAct tiene índice de imagen '?'
    escribir índice de imagen igual para subnodos
en caso contrario
  escribir símbolo '?'
```

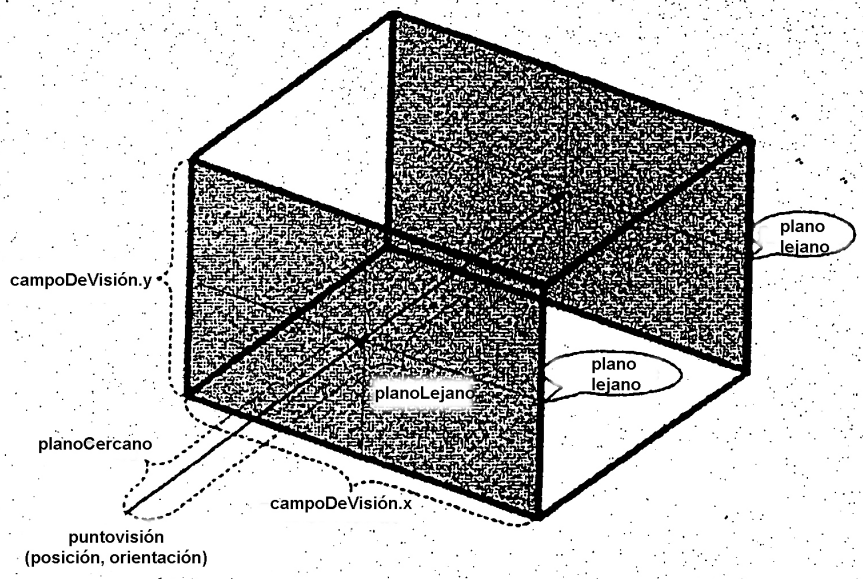
FIG. 26

```
ProfundidadImagen {
        campo   SFVec3f     posición          0 0 10
        campo   SFRotation  orientación       0 0 1 0
        campo   SFVec2f     campoDeVisión     0.785398 0.785398
        campo   SFFloat     planoCercano      10
        campo   SFFloat     planoLejano       100
        campo   SFBool      ortográfico       FALSO
        campo   SFNode      diTextura         NULO
    }

    TexturaSencilla {
        campo   SFNode      textura           NULO
        campo   SFNode      profundidad       NULO
    }

    TexturaPunto {
        campo   SFInt32     ancho             256
        campo   SFInt32     altura            256
        campo   SFInt32     profundidadNbBits  7
        campo   MFInt32     profundidad       []
        campo   MFColor     color             []
    }

  ImagenOctárbol{
        campo   SFInt32     octárbolResolución      256
        campo   MFInt32     octárbol                ""
        campo   MFInt32     voxelImagenÍndice       ""
        campo   MFNode      imágenes                []
    }
    Recientemente archivados
```

FIG. 27


campoDeVisión.y
plano lejano
plano cercano
puntovisión (posición, orientación)
campoDeVisión.x

(a)

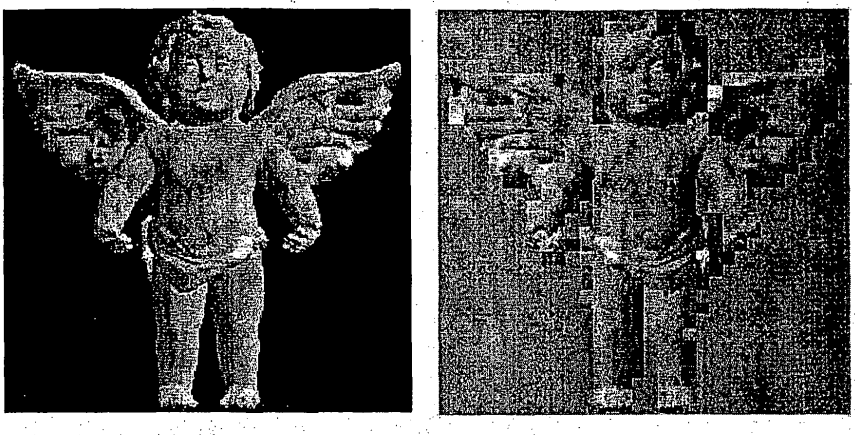

plano lejano
campoDeVisión.y
plano lejano
planoLejano
planoCercano
campoDeVisión.x
puntovisión
(posición, orientación)

(b)

FIG. 28

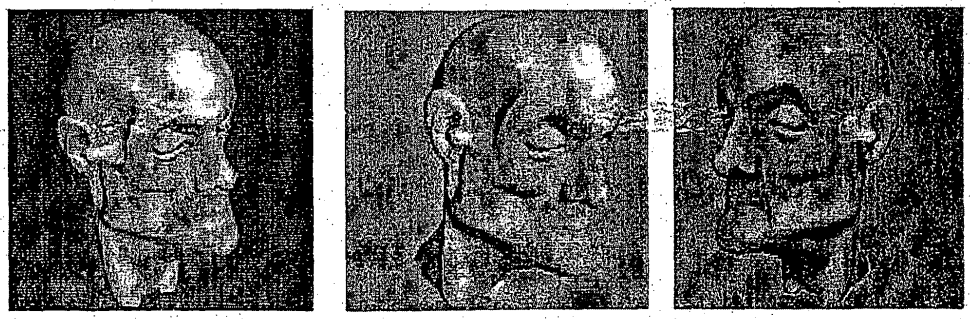
```
// Recorrer todos los puntos de la TexturaSencilla
for y=0 y<altura_imagen.y++
  for x=0 x<ancho_imagen.x++


  // Comprobar si el punto pertenece a la proyección
  // del objeto
  if profundidad_mapa(x,y) |= 0
  {
    // Calcular coordenadas tridimensionales
    // arriba, derecha, dir, centro - cámara
    // vectores de orientación y ubicación
    punto3d = arriba*y + derecha*x +
        dir*profundidad_mapa(x,y) + centro

    // Obtener colores
    color = mapa_color(x,y)

    // Visualizar por medio de OpenGL
    Aplastar punto3d con un radio precalculado
    usando funciones de OpenGL.
  }
```

FIG. 29



(a) (b)

FIG. 30



(a) (b) (c)

FIG. 31

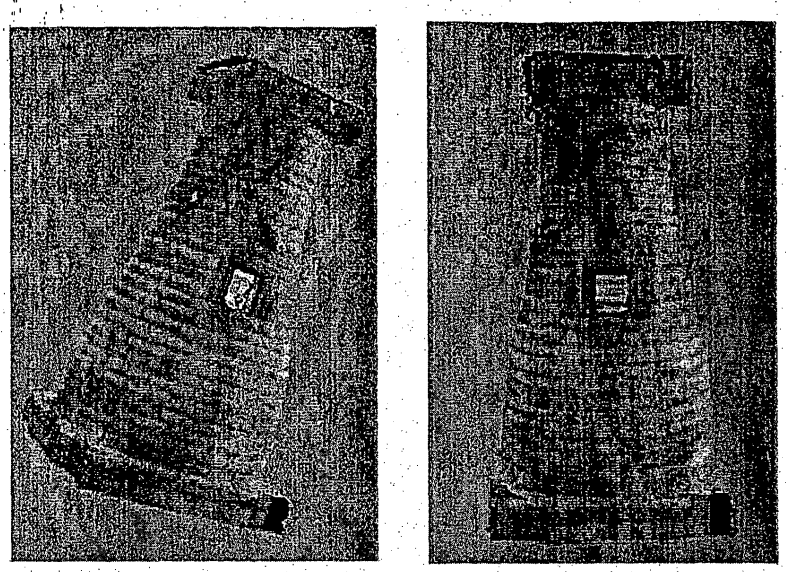


FIG. 32

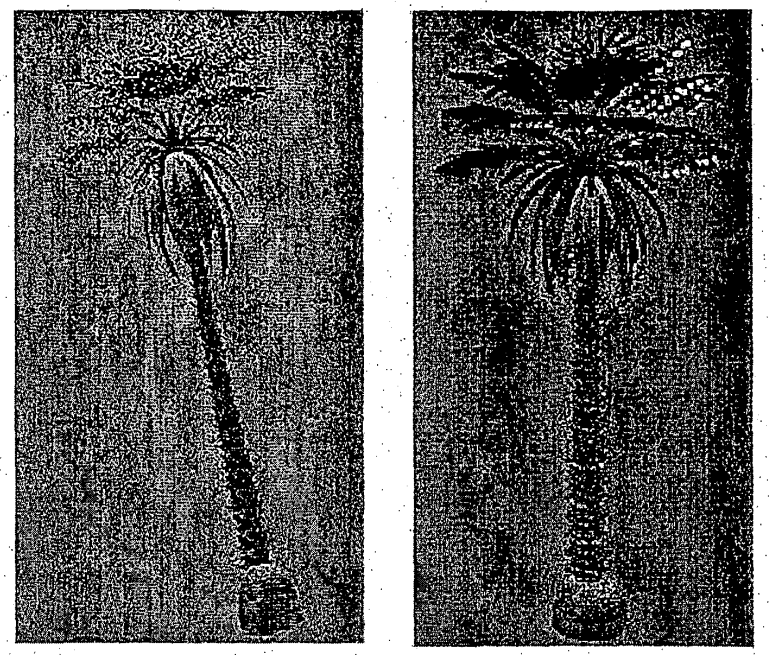


FIG. 33

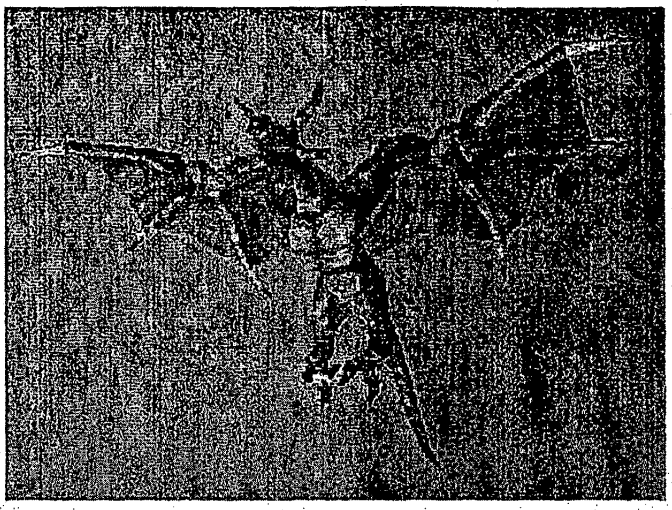


FIG. 34

FIG. 35A

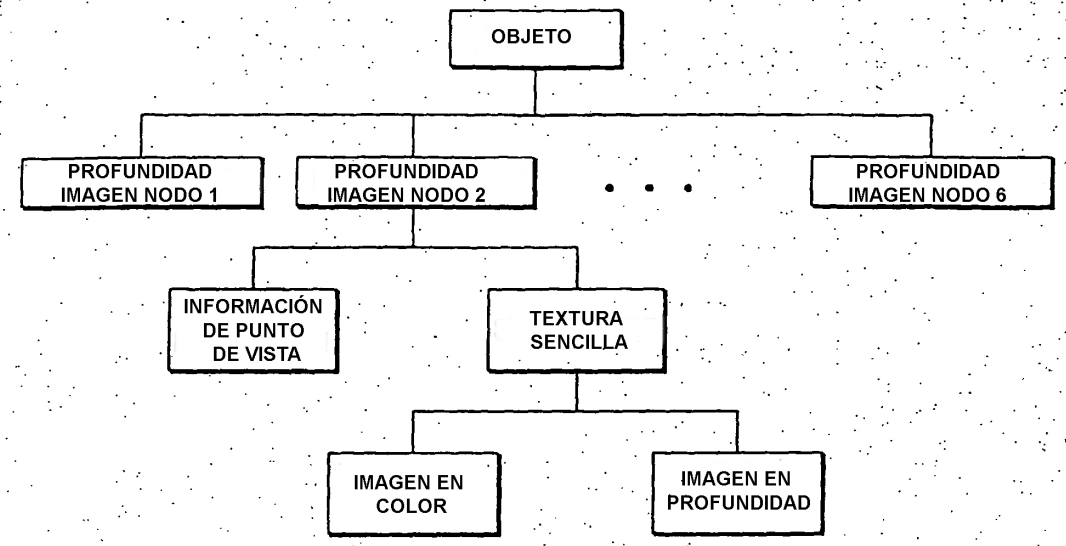


FIG. 35B

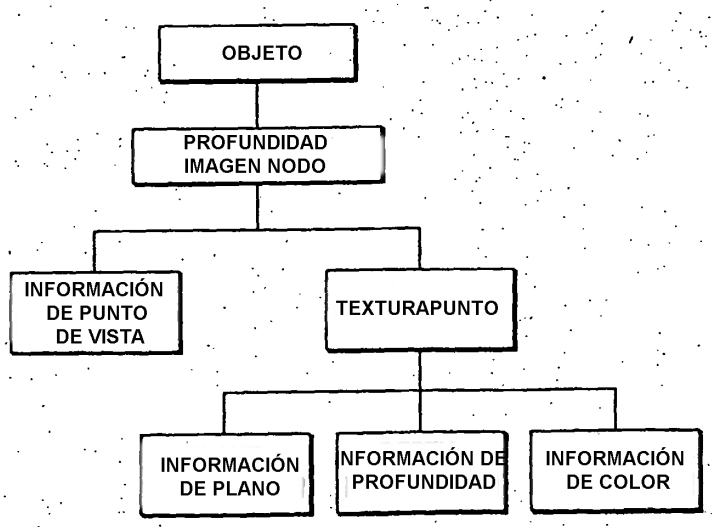
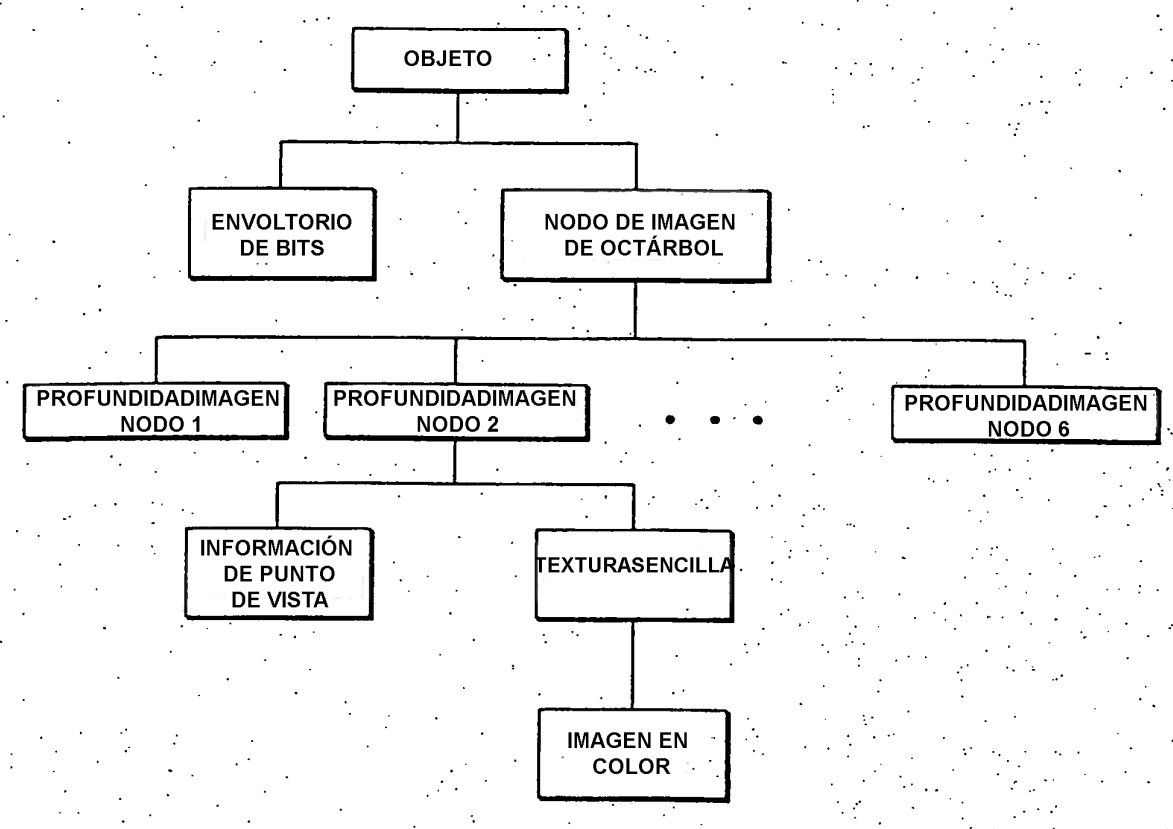


FIG. 36


OBJETO
ENVOLTORIO DE BITS
NODO DE IMAGEN DE OCTÁRBOL
PROFUNDIDADIMAGEN NODO 1
PROFUNDIDADIMAGEN NODO 2
PROFUNDIDADIMAGEN NODO 6
INFORMACIÓN DE PUNTO DE VISTA
TEXTURASENCILLA
IMAGEN EN COLOR