(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 745 143**

(51) Int. CI.:

G06F 17/10 (2006.01)
H03M 7/30 (2006.01)
H04N 19/94 (2014.01)
G10L 19/038 (2013.01)
G10L 19/18 (2013.01)
G10L 19/00 (2013.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(86) Fecha de presentación y número de la solicitud internacional: **12.12.2012 PCT/SE2012/051381**

(87) Fecha y número de publicación internacional: **03.10.2013 WO13147667**

(96) Fecha de presentación y número de la solicitud europea: **12.12.2012 E 12821073 (9)**

(97) Fecha y número de publicación de la concesión europea: **19.06.2019 EP 2831757**

(54) Título: **Cuantificador vectorial**

(30) Prioridad:

29.03.2012 US 201261617151 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
27.02.2020

(73) Titular/es:

TELEFONAKTIEBOLAGET LM ERICSSON (PUBL) (100.0%)
164 83 Stockholm, SE

(72) Inventor/es:

GRANCHAROV, VOLODYA y
JANSSON TOFTGÅRD, TOMAS

(74) Agente/Representante:

LINAGE GONZÁLEZ, Rafael

DESCRIPCIÓN

Cuantificador vectorial.

Campo técnico

La tecnología sugerida se refiere en general a cuantificación vectorial (VQ), y especialmente a la precisión y complejidad computacional de la misma.

Antecedentes

Existen dos clases principales de algoritmos de cuantificación, a saber, cuantificadores escalares (SQs), los cuales procesan un elemento por elemento de vector de entrada, y cuantificadores vectoriales (VQs), los cuales cuantifican un vector de entrada como una unidad (todas las dimensiones vectoriales se cuantifican conjuntamente). A una tasa de bits dada, los VQs son superiores a los SQs, pero a un coste de complejidad computacional y almacenamiento de memoria incrementados.

Supóngase que el vector objetivo a cuantificar sea de M dimensiones: $\mathbf{s} = [s(1), s(2), ..., s(M)]$. El algoritmo de VQ realiza una búsqueda en un libro de códigos (CB) de tamaño K.

$$\{\mathbf{c}_k\}_{k=1}^{K}$$

de vectores de código $\mathbf{c}_k = [c_k(1), c_k(2), ..., c_k(M)]$ de M dimensiones previamente almacenados. Dicha búsqueda devuelve el índice del vector del libro de códigos que proporciona el mejor emparejamiento k^{opt} en base a una medición de distorsión $d(\mathbf{s}, \mathbf{c_k})$. Las ecuaciones (1-2) que siguen describen esta operación, suponiendo que el criterio de búsqueda esté basado en un error cuadrático:

$$k^{opt} = \arg\min_{k} d(\mathbf{s}, \mathbf{c}_k) \qquad (1)$$

$$d(\mathbf{s}, \mathbf{c}_k) = \sum_{m=1}^{M} (s(m) - c_k(m))^2 \qquad (2)$$

El índice óptimo k^{opt} se transmite al descodificador, y se extrae el vector de código correspondiente a partir del CB (CBs idénticos están disponibles tanto en el codificador como en el descodificador), y se usa para reconstruir el vector objetivo. El CB se entrena típicamente fuera de línea y captura las propiedades estadísticas de los datos. En muchos casos, el error cuadrático simple (véase la ecuación (2)) se modifica con pesos, tal como:

$$d(\mathbf{s}, \mathbf{c}_k) = \sum_{m=1}^{M} w(m) \cdot (s(m) - c_k(m))^2 \qquad (3)$$

donde los pesos $w(m)$ son dependientes de la aplicación. Por motivos de simplicidad de la presentación en la presente memoria, solamente se usará el error cuadrático, definido en la ecuación (2), en la descripción que sigue. Sin embargo, debe apreciarse que los principios discutidos en la presente memoria son también válidos cuando se usan criterios más sofisticados, tal como el de la ecuación (3).

Tal y como se puede concluir a partir de la descripción que antecede, la precisión o la calidad de la señal objetivo reconstruida depende del tamaño K del libro de códigos; donde un CB grande conduce a una precisión más alta, y por tanto a una mejor calidad, que un CB más pequeño. Al mismo tiempo, a partir de la ecuación (1), se puede concluir que la principal complejidad computacional está también relacionada con el tamaño del CB, suponiendo que la dimensionalidad del vector sea fijada por la aplicación.

Típicamente, los sistemas de transmisión de audio se construyen bajo las restricciones de complejidad computacional limitada. Es decir, en el peor caso, la complejidad no deberá exceder un determinado nivel L_{MAX} previamente definido. Por ejemplo, la complejidad computacional de un códec de audio se mide típicamente por medio de Millones de Operaciones Ponderadas por Segundo (WMOPS), pero dado que se está considerando un módulo de VQ, la complejidad está relacionada directamente con el tamaño del espacio de búsqueda (tamaño del CB). Un VQ es típicamente el módulo más complejo en un códec, y además, la búsqueda del CB (número de comparaciones con vectores de CB) es lo que hace que el VQ sea tan complejo.

Si un sistema de VQ se usa para cuantificar un vector **S** objetivo cada vez, el espacio K de búsqueda ha de ser optimizado de tal modo que la complejidad no exceda de L_{MAX}. Algunas técnicas de optimización fuera de línea, tal

como un VQ de corte y múltiples etapas, pueden proporcionar una cierta reducción de la complejidad (y del almacenaje), dadas las propiedades del vector **s** y los requisitos de calidad para el vector reconstruido.

Si el sistema de VQ está destinado a cuantificar múltiples vectores:

$$\{\mathbf{s}_n\}_{n=1}^{N}$$

objetivo (de entrada) cada vez, con un número variable de vectores *N*, las técnicas de optimización fuera de línea mencionadas con anterioridad no están capacitadas para mantener las restricciones de complejidad y calidad. En esos casos, la optimización fuera de línea debe encontrar un equilibrio entre los requisitos de contradicción de: A) limitación de la complejidad (= limitación de la búsqueda) cuando un gran número de vectores de entrada han de ser cuantificados simultáneamente, y B) mantenimiento de una alta precisión (= búsqueda en un libro de códigos grande) cuando un bajo número de vectores han de ser cuantificados, lo que no es una tarea sencilla.

El documento de la técnica anterior “Fast codebook search algorithm for unsconstrained vector quantization”, Chen C. Q. et al., divulga un método de cuantificación de vector donde se seleccionan C grupos de candidatos fuera de los T grupos para cada vector de entrada. La búsqueda completa se realiza solamente en esos C grupos de candidatos.

Por otra parte, el documento de la técnica anterior “Application of sorted codebook vector quantization to spectral coding of speech”, Nohammadi H.R.S. et al., divulga un método de cuantificación de vector donde el libro de códigos es clasificado de acuerdo con criterios predefinidos en una etapa de procesamiento previo. Durante la etapa de codificación, se calcula en primer lugar la posición del vector de entrada, y a continuación se lleva a cabo una búsqueda completa en las proximidades con compensación.

Sumario

La tecnología descrita en la presente memoria es aplicable, por ejemplo, para sistemas de compresión/transmisión de audio y de video que llevan a cabo compresión con pérdidas sobre un flujo de entrada, y podría ser descrita según un número de aspectos diferentes. La tecnología descrita en la presente memoria incluye un libro de códigos, el cual está dividido en clases y ordenado, y la clasificación de un vector **s** objetivo de entrada a ser cuantificado en una de dichas clases. La tecnología descrita en la presente memoria permite que la clase de vectores de código, en el libro de códigos, que comprende el conjunto más probable de vectores de código candidatos con relación al vector **s** de entrada, sea la primera buscada de las clases en el libro de códigos. De ese modo, el vector de código de mejor emparejamiento para el vector **s** de entrada puede ser encontrado pronto en una búsqueda, y se puede reducir la complejidad computacional.

La invención es tal y como se define en las reivindicaciones independientes. Las realizaciones preferidas están especificadas en las reivindicaciones dependientes.

El tamaño de una región de búsqueda en el libro de códigos, en la que se lleva a cabo una búsqueda, puede estar adaptado en base a un número de vectores objetivo de entrada y a una restricción de complejidad máxima. El número de vectores objetivo de entrada por unidad de codificación puede ser variable. Además, la restricción de complejidad máxima puede ser establecida dinámicamente. Además, se podría realizar una búsqueda en el libro de códigos en el espacio de búsqueda determinado, empezando en un punto de inicio determinado, donde la búsqueda proporciona un mejor emparejamiento para el vector **s** objetivo de entrada. Por “mejor emparejamiento” se pretende indicar en la presente memoria el emparejamiento más cercano, la distancia más corta, con relación a una medición de distancia entre el vector objetivo de entrada y un vector candidato en el libro de códigos, es decir, un mejor emparejamiento es un vector de código que tiene la distancia más corta al vector objetivo de entrada, de acuerdo a la medición de distancia.

Breve descripción de los dibujos

La tecnología sugerida va a ser descrita ahora con mayor detalle por medio de ejemplos de realización y con referencia a los dibujos que se acompañan, en los que:

Las Figuras 1a y 1b muestran la estructura de un CB ordenado conforme a una solución descrita en la presente memoria. La búsqueda se inicia a partir del punto 0 o del punto 1 hacia el otro extremo del CB.

La Figura 2 muestra un ejemplo de estructura de CB que aprovecha simetría. Solamente vectores de código de C_0 y C_1 están almacenados en una memoria ROM.

La Figura 3 ilustra un ejemplo de unidad funcional para determinar una clase óptima para un vector **s** de entrada por comparación del vector **s** de entrada con cada uno de un número de centroides (C_0, C_1, $C_{1,\text{flip}}$, $C_{0,\text{flip}}$), cada uno de ellos asociado a una clase en un libro de códigos.

La Figura 4 ilustra una unidad funcional para determinar un tamaño de una región de búsqueda en un libro de códigos en base a un número de picos espectrales detectados en una trama actual, y posiblemente en base a la

tasa de bits del códec.

La Figura 5 es una tabla que ilustra el hecho de que una región de búsqueda se incrementa según se reduce el número de picos por trama. En el ejemplo, con 17 picos (= 17 vectores de entrada), la búsqueda se realiza solamente en un CB de 7 bits (definido de moldo que sea el espacio mínimo de búsqueda en este ejemplo), pero con 8 picos o menos, la búsqueda se realiza en un CB de 8 bits (espacio máximo de búsqueda), dado que esto puede ser “permitido” bajo la restricción máxima de complejidad.

Las Figuras 6a-d muestran ejemplos de diferentes regiones de búsqueda.

La Figura 7 ilustra el hecho de que la complejidad L_{MAX} permitida puede ser indicada al sistema a partir de una entidad externa. El parámetro L_{MAX} podría estar basado, por ejemplo, en una carga de CPU, o un estado de batería.

La Figura 8 es un diagrama de flujo que ilustra las acciones en un procedimiento para crear un libro de códigos, CB, para que sea usado en la tecnología sugerida.

Las Figuras 9a-c son diagramas de flujo que ilustran acciones en procedimientos para cuantificación vectorial, VQ, conforme a ejemplos de la tecnología sugerida en la presente memoria.

La Figura 10 es un diagrama de bloques que ilustra un cuantificador vectorial, conforme a un ejemplo de la tecnología sugerida en la presente memoria.

La Figura 11 es un diagrama de bloques que ilustra un códec que comprende un cuantificador vectorial, conforme a un ejemplo de la tecnología sugerida en la presente memoria.

La Figura 12 es un diagrama de bloques que ilustra una disposición para cuantificación vectorial, conforme a un ejemplo de la tecnología sugerida en la presente memoria.

Descripción detallada

Dicho en pocas palabras, la solución descrita en la presente memoria se refiere a adaptar dinámicamente el espacio de búsqueda de un VQ, de tal modo que, para cualquier número de vectores objetivo (de entrada) por bloque o intervalo de tiempo, se logre una cuantificación de alta precisión, y por tanto de alta calidad, dentro de una restricción de complejidad dada. Es decir, no se deben infringir los requisitos de complejidad computacional (véase L_{max}). Esto se consigue en virtud de que la búsqueda se realiza en un CB especial clasificado y ordenado. El punto de inicio en el espacio de búsqueda para cada vector objetivo se basa en un procedimiento de clasificación, y el tamaño del espacio de búsqueda se incrementa o se reduce, en base al número de vectores objetivo. El algoritmo de VQ descrito en la presente memoria puede ser mencionado como una “herramienta” para compresión de datos, independiente de lo que sean los datos, es decir, los datos podrían ser, por ejemplo, video y/o audio. En esta descripción, el VQ se describe en el contexto de un códec de audio, pero el concepto descrito en la presente memoria no se limita a codecs de audio. Por ejemplo, también se podría implementar en codecs de video.

El algoritmo descrito en la presente memoria se basa en un CB diseñado de forma especial. Algunas variantes de dicho libro de códigos van a ser descritas con mayor detalle más adelante. En primer lugar se va a describir un caso básico, y más adelante se va a discutir un esquema más avanzado. Los vectores de código del CB pueden ser organizados conforme a la solución descrita en la presente memoria en modo de fuera de línea.

Con el fin de crear una versión básica de del CB ventajoso especialmente diseñado, los vectores de código de un CB se dividen en dos clases, indicadas aquí como C_0 y C_1 (esta notación será usada tanto para los nombres de las clases, como para los centroides correspondientes, véase la Figura 1). Para dividir los datos en dos clases se puede usar lo que se conoce como algoritmo K-means (algoritmo de Lloyd Generalizado). Éste constituye una técnica bien conocida, que toma un conjunto completo de datos como entrada, y el número deseado de clases, y presenta a la salida los centroides del número deseado de clases. Por ejemplo, el algoritmo presenta a la salida 2 vectores de centroide si el número deseado de clases ha sido indicado como 2. Obsérvese que estos centroides, cuando se usa un algoritmo de K-means, son vectores de la misma dimensión que los vectores del conjunto de datos, pero no pertenecen al conjunto de datos. Es decir, los vectores de centroide están fuera del CB y no necesitan coincidir con otros vectores de código existentes. Mediante “centroide” se indica en la presente memoria, en general, un vector de referencia que representa una clase de vectores.

Todos los vectores de código del CB son clasificados a continuación conforme a una medición de distorsión, por ejemplo como la definida en la ecuación (4):

$$d(\mathbf{c}_k, C_0, C_1) = \sum_{m=1}^{M} \left(c_k(m) - C_0(m)\right)^2 - \sum_{m=1}^{M} \left(c_k(m) - C_1(m)\right)^2 \quad (4)$$

La medición de distorsión anterior da como resultado, o adopta, grandes valores negativos para vectores de código cercanos a C_0 y grandes valores positivos para vectores de código cercanos a C_1. Los vectores de código que están

distanciados equidistantemente de los centroides (C_0 y C_1) de las dos clases producen una distorsión medida que está próxima a cero. En el CB, los vectores de código están ordenados, por ejemplo, en virtud de la medición de distorsión creciente, según se ha ilustrado en las Figuras 1a y b.

Cada vector objetivo de entrada se compara con los dos centroides (el centroide respectivo de las dos clases) y se asigna a, dependiendo del resultado, es decir si se concluye o se determina a quién pertenece, ya sea a la clase C_0 o ya sea a la clase C_1. En base a esa clasificación, el punto de inicio de la búsqueda se selecciona de modo que sea el punto más superior (Figura 1a) o el punto 0 más a la izquierda (Figura 1b) (cuando el vector objetivo pertenece a la clase C_0), o bien el punto más inferior (Figura 1a) o el punto 1 más a la derecha (Figura 1b) (cuando el vector objetivo pertenece a la clase C_1). Ahora, el tamaño del espacio de búsqueda deberá hacerse dependiente del número de vectores *N* objetivos de entrada por bloque o segmento/intervalo de tiempo. Si se redefine el espacio K de búsqueda de modo que no sea el tamaño del CB completo, sino que sea variable, el concepto detrás de la adaptación descrita en el presente documento puede ser definido en la ecuación (5):

$$N \times K \approx const \qquad (5)$$

En otra palabras, *#Quantizers* x *#Operations_per_Quantizer* ≈ const., donde “Quantizer” puede ser considerado como el algoritmo que mapea un vector de entrada respecto a uno de los vectores de código.

En la presente memoria, como ejemplo, el VQ se describe en un contexto de un códec de transformación que codifica picos espectrales, o de manera estricta, las regiones en torno a picos espectrales. En el contexto de un códec de ese tipo, un vector objetivo de entrada puede reflejar un pico (región) espectral de un segmento de la señal de audio que está siendo procesada. El número de picos espectrales en el espectro de la señal de un segmento de tiempo, por ejemplo 30 ms, de una señal de audio, depende de las propiedades espectrales de la señal de audio en ese segmento de tiempo. Puesto que las propiedades espectrales de una señal de audio pueden variar con el tiempo y son diferentes, por ejemplo, para diferentes tipos de audio, el número de picos espectrales puede variar entre diferentes segmentos de tiempo y entre diferentes señales de audio. De ese modo, cuando se usa un codificador de transformación que codifica regiones de picos espectrales, el número de vectores de entrada, por bloque o por segmento de tiempo, respecto al VQ variará. En los ejemplos de la presente memoria, el número máximo de vectores de entrada, correspondientes a un número de picos espectrales en un segmento de tiempo de una señal de audio, es 17. Sin embargo, este número es solamente un ejemplo, y no debe ser interpretado como limitación de la solución en general.

Efectivamente, el esquema descrito con anterioridad mantiene el número de operaciones requeridas para el VQ en un estrecho intervalo (o casi constante); es decir, cuando el número de VQs se incrementa, es decir el número de vectores objetivo de entrada se incrementa, el número de operaciones por VQ se reduce (el tamaño del espacio de búsqueda disminuye/solamente se busca en parte del CB), de tal modo que los requisitos de complejidad (es decir, las restricciones) no se infringen. Con un descenso de *N*, el espacio *K* de búsqueda se puede aumentar, como máximo hasta el tamaño del CB completo, lo que conduce a una precisión más alta y por tanto a una mayor calidad del vector reconstruido. La precisión de un cuantificador vectorial puede ser medida como error cuadrático entre una señal original y los datos reconstruidos correspondientes.

De esta manera, el libro de códigos del VQ no necesita estar diseñado para el escenario del peor caso (es decir, el número máximo de vectores objetivo de entrada). En cambio, podría estar diseñado, por ejemplo, para un escenario del mejor caso, comprendiendo de ese modo más vectores de código de los que posiblemente podrían ser investigados para el número máximo de vectores objetivo de entrada dentro de la restricción L_{MAX} de máxima complejidad. El requisito de complejidad máxima se cumplirá en la medida en que la magnitud de la búsqueda, es decir el tamaño del espacio de búsqueda, en el CB dependa del número de vectores objetivo de entrada. Sin embargo, si esto se hiciera “a ciegas”, por ejemplo sin el CB sugerido en la presente memoria, la calidad de la cuantificación se vería considerablemente afectada, puesto que no habría manera de saber dónde está situado el vector de “mejor emparejamiento” en el CB, o si este vector de mejor emparejamiento está ubicado en una parte del libro de códigos que será investigada cuando se reduzca el espacio de búsqueda. Este problema se resuelve mediante el diseño especial del libro de códigos, que se describe en la presente memoria. Debe apreciarse que el diseño de CB descrito en la presente memoria es beneficioso para aplicaciones en que el número de vectores de entrada, por unidad de codificación, sea constante.

Ejemplo de realización: VQ restringido en regiones de picos espectrales

Un conjunto de vectores **s** espectrales representan regiones de pico espectral en una codificación de audio en el dominio de transformación, por ejemplo coeficientes de transformación en las cercanías de picos de MDCT. De ese modo, en este contexto, el número de vectores objetivo varía con el tiempo, dado que el número de picos espectrales varía de un bloque de tiempo a otro.

En este tipo de aplicación (codificación de región de pico), los vectores **S** objetivo presentan ciertas simetrías que pueden ser usadas para optimizar aún más el CB. Por ejemplo, los coeficientes de transformación a ambos lados de un pico espectral tienen estadísticas similares. Si se supone que los vectores **S** objetivo están centrados en la posición del pico, la simetría descrita con anterioridad permite añadir otra estructura en el CB ordenado de las

Figuras 1a y 1b. La estructura de ese nuevo CB, adicionalmente mejorado, ha sido ilustrada en la Figura 2. La Figura 2 ilustra un CB donde los vectores de código de la parte de la izquierda están almacenados en una memoria, tal como una Memoria de Sólo Lectura. Sin embargo, no existen vectores de código pre-almacenados para el lado de la derecha del CB, es decir, para las clases $C_{1,flip}$ y $C_{0,flip}$. Los vectores de código de esas clases son versiones invertidas de los vectores de código del lado derecho del CB. De ese modo, cuando se realiza una búsqueda en las clases $C_{0,flip}$ y $C_{1,flip}$, la búsqueda se realiza sobre los vectores de código del lado de la izquierda, los cuales son almacenados en memoria, pero con los elementos de los vectores de código invertidos en torno al centro, de tal modo que los vectores de código $\mathbf{c}_{k,flip}$ vienen dados por la ecuación (6):

$$\mathbf{c}_{k,flip} = \left[c_k(M) \quad c_k(M-1) \quad \ldots \quad c_k(1)\right], \qquad (6)$$

donde $c_k(m)$ son los elementos vectoriales de la clase C_j correspondiente en el CB almacenado (es decir, C_0 o C_1). Es decir, si los elementos de un determinado vector de código en C_0 son [C_{01}, C_{02}, C_{03}, C_{04}}, los elementos de un vector de código correspondiente en $C_{0,flip}$ son {C_{04}, C_{03}, C_{02}, C_{01}}.

Cuando se usa un CB como el que se ha ilustrado en la Figura 2, se compara un vector objetivo de entrada con cuatro centroides y se asigna a una clase con vistas a determinar un punto de inicio para la búsqueda, es decir, la clase óptima para el vector objetivo de entrada se determina por comparación del vector **S** de entrada con cada uno de los centroides {C_0, C_1, $C_{0,flip}$, $C_{1,flip}$}. Esto se ha ilustrado en la Figura 3, donde un vector **S** objetivo se introduce en una unidad 302 de asignación de clase, la cual proporciona un indicador de clase, C_j, como salida. Los centroides $C_{1,flip}$ y $C_{0,flip}$ no están almacenados en ninguna tabla, sino que son "creados" al invertir los elementos de los centroides C_0 y C_1. Los elementos no necesitan ser literalmente invertidos, sino que una operación de búsqueda modificada puede leer los elementos de C0 y C1 en el orden inverso cuando se lee C0,flip y C1,flip, es decir, ambos centroides y vectores del libro de códigos. De esta manera, el CB puede ser ampliado de modo que comprenda el doble del número de vectores de código, en comparación con lo que esté físicamente almacenado en el CB, lo que significa ahorro de recursos de memoria. Esto es posible debido a que se aprovecha la simetría de las regiones de pico, según se ha descrito con anterioridad. Más específicamente, la solución se basa en la observación de que la simetría puede ser aprovechada de modo que un vector de código válido invertido sea también un vector de código válido.

La región de búsqueda está adaptada al número de picos espectrales, el cual corresponde al número de vectores objetivo de entrada. Esto ha sido ejemplificado en la Figura 4, la cual muestra una unidad 402 funcional que adopta un indicador de un número de picos/vectores como entrada, y que genera un indicador de una región de búsqueda como salida. La Figura 4 ilustra además el hecho de que, por ejemplo, la tasa de bits de un códec que aplica el VQ, podría ser tomada en consideración cuando se determina la región de búsqueda (espacio de búsqueda). Por ejemplo, pueden existir requisitos de calidad diferentes para diferentes tasas de bits; por ejemplo, cuánto más alta sea la tasa de bits, más alta será la calidad esperada. Además, la complejidad máxima permitida puede cambiar con la tasa de bits, por ejemplo debido a que se activan diferentes módulos de tasas de bits diferentes en un códec, las cuales no son igual de complejas, es decir, la complejidad restante permitida para el VQ, a partir de una restricción de complejidad máxima sobre el procedimiento de codificación global, podría no ser la misma. Es decir, la información de tasa de bits refleja cambios de calidad y de requisitos de complejidad, los cuales pueden ser tomados en consideración en la cuantificación de vector.

La tabla de la Figura 5 ilustra cómo se adapta la región de búsqueda al número de picos. En la Figura 5, la región de búsqueda ha sido indicada como el número de coeficientes (vectores de código) en la búsqueda por vector de entrada. Los números de la tabla en la Figura 5 se deducen bajo la suposición de que el CB de la Figura 2 comprende cuatro segmentos de 7 bits (cuatro segmentos con 128 vectores de código cada uno) de los que dos son "normales" o "físicos" y dos son "invertidos" o "virtuales").

La lógica tras la tabla de la Figura 5 consiste en que, cuando existe un pico menos para codificar, por ejemplo 16 en vez de 17, las 128 comparaciones "ahorradas" pueden ser distribuidas entre los picos restantes. En algún punto, la longitud de la búsqueda se satura, debido a que alcanza el tamaño físico del CB. En el ejemplo ilustrado en la Figura 5, este punto se alcanza cuando el número de picos es 8 o menos. Es decir, en el ejemplo ilustrado en la Figura 5, cuando el número de picos es 8 o menos, se podría realizar una búsqueda completa para todos los vectores objetivo de entrada (es decir, los picos) sin que se alcance la máxima complejidad permitida.

Ejemplos del procedimiento de búsqueda han sido ilustrados en las Figuras 6a-d. En la práctica, es el mismo tipo de búsqueda que el descrito con anterioridad junto con las Figuras 1a-b, pero con clasificación adicional en segmentos/clases de CB "normales" e "invertidos".

En el ejemplo mostrado en la Figura 6a, el vector **S** de entrada pertenece a la clase C1 (posición indicada con flecha hacia abajo). El espacio de búsqueda está entonces limitado a tamaños entre la clase C1 solamente, y al espacio conjunto de las clases C_0 y C_1. En la Figura 6a, esto ha sido ilustrado con tres flechas discontinuas que indican espacios de búsqueda de diferentes tamaños. La Figura 6b ilustra el caso en que un vector de entrada pertenece a la clase C_0 (posición indicada con flecha descendente), en cuyo caso la búsqueda tiene un punto de inicio diferente.

De forma análoga, según se ha ilustrado en las Figuras 6c-d, la búsqueda puede ser realizada en las clases $C_{1,flip}$ y/o $C_{0,flip}$ si el vector de entrada pertenece a una de esas clases. No se realiza ninguna búsqueda en el espacio conjunto de C_1 y $C_{1,flip}$. La razón para hacer esto es que el espacio conjunto de una clase regular y una clase invertida, no corresponde a un conjunto de datos reales (su estadística no corresponde a estadísticas de datos reales). Por lo tanto, no es muy probable que se pueda hallar un buen emparejamiento para un vector de entrada en ese espacio.

Sistema de comunicación con control externo de complejidad máxima permitida

El concepto de un VQ que tenga una complejidad que se ajuste dinámicamente al número de vectores N objetivo, puede ser extendido al caso en que el límite de complejidad no esté predeterminado, sino que puede variar, por ejemplo, en base a algún criterio, y ser señalizado para el VQ y/o para la entidad en la se aplica el VQ. Esto ha sido ilustrado en el diagrama esquemático de bloques de la Figura 7, el cual muestra una unidad 702 funcional que toma un indicador o el número de picos/vectores como entrada, y que toma además una restricción L_{MAX} de complejidad como entrada. La unidad 702 funcional proporciona un indicador de un espacio/una región de búsqueda del CB, véanse las flechas discontinuas de las Figuras 6a-d.

El algoritmo de VQ presentado en la presente memoria con complejidad ajustable, proporciona el equilibrio óptimo entre precisión de cuantificación (es decir, calidad) y mantenimiento de complejidad computacional por debajo de un umbral predefinido.

....

Ejemplo de procedimiento para conseguir estructura de CB

Un ejemplo de procedimiento para diseñar u organizar un CB para su uso en un VQ va a ser descrito a continuación, con referencia a la Figura 8. El procedimiento está destinado a generar un CB para su uso en un VQ que proporciona cuantificación en un codificador de audio de transformación, tal como, por ejemplo, un codificador de MDCT.

El procedimiento que se describe en lo que sigue se refiere a las partes de un procedimiento de creación de CB que se desvían de, y/o son adicionales a, una creación u organización de CB de VQ convencional.

El CB se divide en clases en una acción 802, por ejemplo mediante el uso de lo que se conoce como algoritmo K-means, según se ha descrito con anterioridad. Los vectores de código del CB son clasificados a continuación en el CB en base a una medición de distorsión, por ejemplo como la que se ha descrito en la ecuación (4). La medición de distorsión para cada vector de código depende de una relación entre el vector de código y centroides que representan cada clase del CB, según se ha descrito con anterioridad.

Esta organización del CB permite la adaptación del espacio de búsqueda, y por lo tanto de la complejidad de búsqueda en VQ, a una calidad de VQ altamente conservada (por ejemplo, la calidad de los vectores objetivo reconstruidos).

Ejemplo de procedimiento de VQ

Un ejemplo de procedimiento en un cuantificador vectorial (VQ) va a ser descrito a continuación, con referencia a la Figura 9a. El procedimiento es adecuado para su uso en un codificador de audio de transformación, tal como por ejemplo un codificador de MDCT, que codifica por ejemplo regiones de pico espectral. La señal de audio podría comprender, por ejemplo, habla y/o música.

Un número N de vectores objetivo de entrada son recibidos por el VQ, según se ha descrito anteriormente. A continuación se van a describir las acciones asociadas a uno de los vectores objetivo de entrada, por motivos de simplicidad.

Un vector **s** objetivo de entrada se compara con un número de vectores de código de los que cada uno representa una clase de CB (véanse las clases C_0 y C_1 etc., descritas anteriormente), preferiblemente el centroide de cada clase. La comparación ha sido ilustrada como acción 902 en las Figuras 9a-c. La acción 902 podría estar considerada alternativamente como integrada con una acción 904, ilustrada en la Figura 9c. Dependiendo del resultado de las comparaciones, se asigna al vector **s** objetivo de entrada una de las clases, o secciones, del CB, en una acción 904. El vector **s** objetivo de entrada se asigna a, o se concluye que pertenece a, la clase respecto a la que tiene la distancia más corta, es decir a la que es más similar, de acuerdo con alguna medición de distancia (medición de error). El punto de inicio de la búsqueda en el CB se determina en una acción 906, en base a la asignación de clase o a la medición de distancia.

Se puede realizar una búsqueda en el libro de códigos en una acción 910. La búsqueda se inicia en el punto de inicio seleccionado, y se realiza a través de un espacio de búsqueda, que puede ser de un tamaño determinado, que comprende una o más clases, o partes de las mismas. Debido al CB diseñado y organizado ventajosamente, la probabilidad de que el mejor emparejamiento, de todos los vectores de código candidatos dentro del CB global, para el vector **s** objetivo de entrada, se halle dentro del espacio de búsqueda, es muy alta, incluso cuando el espacio de

búsqueda está limitado, por ejemplo, a la mitad del CB. En un caso en que el espacio de búsqueda podría comprender el libro de códigos completo, el vector de código de mejor emparejamiento podría ser halla pronto durante la búsqueda cuando se inicia la búsqueda en el punto de inicio determinado.

Cuando se encuentra el mejor emparejamiento dentro del espacio de búsqueda determinado, se proporciona el índice del vector de código de mejor emparejamiento, como resultado desde el VQ, en una acción 912, por ejemplo para su uso en un descodificador de audio.

Además, el tamaño del espacio de búsqueda puede ser determinado en una acción 908 ilustrada en la Figura 9c. El espacio de búsqueda puede ser descrito como el número de vectores de código en el CB que deberían ser evaluados durante la búsqueda para el mejor emparejamiento para el vector **s** objetivo de entrada. El tamaño del espacio de búsqueda se determina en base al número de vectores objetivo de entrada y a una restricción sobre la complejidad L_{MAX} computacional. De ese modo, el tamaño del espacio de búsqueda puede ser determinado, por ejemplo, una vez para cada bloque de codificación, o en algún otro intervalo de tiempo, dependiendo por ejemplo de las características de la señal a ser cuantificada y/o de los atributos de un códec. Si el número de vectores objetivo de entrada y la restricción L_{MAX} son constantes o semi-constantes en el tiempo, el tamaño del espacio de búsqueda puede mantenerse también constante a través del tiempo correspondiente.

A continuación, ejemplo de disposición de VQ

En lo que sigue, se va a describir un ejemplo de disposición de VQ adecuada para su uso en un codificador/códec de transformación con referencia a la Figura 10. El códec de transformación podría ser, por ejemplo, un códec de MDCT. El VQ está adaptado para llevar a cabo las acciones del procedimiento descrito en lo que antecede.

El VQ 1001 ha sido ilustrado de modo que comunica con otras entidades (por ejemplo, un códec de audio) a través de una unidad 1002 de comunicación. El VQ puede comprender además otras unidades 1016 funcionales, tal como, por ejemplo, unidades funcionales que proporcionen funciones regulares, y puede comprender además una o más unidades 1014 de almacenamiento.

El VQ 1001 podría ser implementado, por ejemplo, mediante uno o más de entre un procesador o un microprocesador y software adecuado con almacenaje adecuado por lo tanto, un Dispositivo Lógico Programable (PLD) u otro(s) componente(s) y/o circuito(s) electrónico(s).

La unidad 1002 de comunicación se supone que comprende unidades funcionales para obtener los parámetros adecuados, tal como vectores objetivo de entrada y L_{MAX} proporcionados, por ejemplo, desde una entidad de codificación.

El VQ puede comprender una unidad 1004 de comparación, que esté adaptada para comparar un vector **s** objetivo de entrada con vectores que representan cada clase del CB, por ejemplo el vector de centroide de cada clase. Además, el VQ puede comprender una unidad 1006 de asignación, la cual está adaptada para asignar una clase al vector **s** objetivo de entrada (o asignar el vector **s** a una clase), es decir, concluir a qué clase pertenece el vector, en base a la comparación. Además, el VQ puede comprender una unidad 1008 de determinación, adaptada para determinar un punto de inicio adecuado para una búsqueda en el CB, en base a la clase asignada al vector **s**. La unidad de determinación puede estar además adaptada para determinar el tamaño de un espacio de búsqueda en el CB, en base por ejemplo a un número de vectores objetivo de entrada recibidos y a una limitación de complejidad computacional.

Además, el VQ puede comprender una unidad 1010 de búsqueda, que está adaptada para llevar a cabo una búsqueda en el CB, empezando en el punto de inicio determinado y buscando el espacio de búsqueda determinado. La búsqueda podrá dar como resultado uno o más índices de CB que apunten al vector de código que se empareje mejor con el vector **s** objetivo de entrada. El VQ puede comprender además una unidad 1012 de provisión, que esté adaptada para proporcionar dicho(s) índice o índices a otra entidad, por ejemplo a (o para su uso por) un códec de transformación.

Ejemplo de disposición

La Figura 12 muestra esquemáticamente una realización de una disposición 1200 adecuada para su uso, por ejemplo, en un descodificador de audio de transformación, la cual puede ser también una forma alternativa de divulgación de una realización del VQ ilustrado en la Figura 5. En este caso está comprendida en la disposición 1200 una unidad 1206 de procesamiento, por ejemplo con un DSP (Procesador Digital de Señal). La unidad 1206 de procesamiento puede ser una única unidad o una pluralidad de unidades para llevar a cabo diferentes etapas de procedimientos descritos en la presente memoria. La disposición 1200 puede comprender también la unidad 1202 de entrada para recibir señales, tal como vectores objetivo de entrada e indicadores de, por ejemplo, una tasa de bits y/o una limitación de complejidad; y además, la unidad 1204 de salida para señal(es) de salida, tal como los índices de CB para los vectores de código de mejor emparejamiento. La unidad 1202 de entrada y la unidad 1204 de salida pueden estar dispuestas como solo una en el hardware de la disposición.

Además, la disposición 1200 comprende al menos un producto 1208 de programa informático en forma de memoria no volátil, por ejemplo una EEPROM, una memoria flash y un disco duro. El producto 1208 de programa informático comprende un programa informático 1210, el cual comprende medios de código, que cuando se ejecutan en la unidad 1206 de procesamiento en la disposición 1200, hacen que la disposición lleve a cabo las acciones de un procedimiento descrito en lo que antecede junto con las Figuras 9a-c.

Con ello, en los ejemplos de realización descritos, el medio de código en el programa informático 1210 de la disposición 1200 puede comprender un módulo 1210a de comparación para comparar un vector objetivo de entrada con centroides de clase de un CB. El programa informático puede comprender un módulo 1210b de asignación para asignar una clase al vector objetivo de entrada. El programa informático 1210 puede comprender además una unidad 1210c de determinación para determinar un punto de inicio para una búsqueda en el CB; y además, para determinar un espacio o región de búsqueda en base a parámetros de entrada. El programa informático 1210 puede comprender además una unidad 1210d de búsqueda para buscar el CB conforme a lo anterior. Además, el programa informático 1210 puede comprender un módulo 1210e de provisión, para proporcionar índices, los cuales son la señal de salida de la búsqueda para otras entidades.

El programa informático 1210 tiene forma de código de programa informático estructurado en módulos de programa informático. Los módulos 1210a-e pueden realizar esencialmente las acciones del flujo ilustrado en cualquiera de las Figuras 9a-c para emular al menos parte del VQ 1001 ilustrado en la Figura 10. En otras palabras, cuando se ejecutan los diferentes módulos 1210a-d en la unidad 1206 de procesamiento, éstos corresponden al menos a las unidades 1004-1012 de la Figura 10.

Aunque los medios de código en la realización divulgada anteriormente junto con la Figura 12 están implementados a modo de módulos de programa informático que cuando se ejecutan en la unidad de procesamiento hacen que la disposición y/o el codificador de audio de transformación lleven a cabo etapas descritas con anterioridad en combinación con las Figuras mencionadas con anterioridad, al menos uno de los medios de código puede estar implementado, en realizaciones alternativas, al menos parcialmente a modo de circuitos de hardware.

Mientras que la tecnología sugerida ha sido descrita con referencia a ejemplos de realización específicos, la descripción está destinada en general solamente a ilustrar el concepto y no debe ser tomada como limitativa del alcance de la tecnología descrita en la presente memoria. Las diferentes características de los ejemplos de realización anteriores pueden ser combinadas de formas diferentes conforme a las necesidades, requisitos o preferencias.

La solución descrita con anterioridad puede ser usada siempre que se apliquen VQs, por ejemplo en codecs de dispositivos tales como terminales móviles, tabletas, ordenadores, teléfonos inteligentes, etc.

Debe entenderse que la opción de unidades o módulos de interacción, así como la denominación de las unidades se proporcionan solamente con fines de ejemplificación, y los nodos adecuados para ejecutar cualquiera de los métodos descritos con anterioridad pueden estar configurados según una pluralidad de formas alternativas de modo que estén capacitados para ejecutar las acciones de proceso sugeridas.

Las funciones de los diversos elementos que incluyen bloques funcionales, que incluyen aunque sin limitación los etiquetados o descritos como “unidad funcional”, “procesador” o “controlador”, pueden ser proporcionados mediante el uso de hardware tal como hardware de circuito y/o hardware capacitado para ejecutar software en forma de instrucciones codificadas almacenadas en un medio legible con ordenador. De ese modo, tales funciones y los bloques funcionales ilustrados, deben ser entendidos como que son implementados ya sea en hardware y/o ya sea implementados en ordenador, y por tanto implementados con máquina.

En términos de implementación de hardware, los bloques funcionales pueden incluir o abarcar, sin limitación, hardware de procesador de señal digital (DSP), procesador de conjunto de instrucciones reducido, circuitería de hardware (por ejemplo, digital o analógica) incluyendo, aunque sin limitación, circuito(s) integrado(s) específico(s) de la aplicación (ASIC) y (donde sea apropiado) máquinas de estado capaces de llevar a cabo tales funciones.

Abreviaciones

SQ Cuantificación Escalar

VQ Cuantificación Vectorial

CB Libro de Códigos

WMOPS Millones de Operaciones Ponderadas por Segundo

MDCT Transformada Coseno Discreta Modificada

REIVINDICACIONES

1.- Método para codificación de región de pico realizada por un códec (1100) de transformación que comprende un Cuantificador Vectorial, comprendiendo el método:

operar un procesador (1206) del códec (1100) de transformación que ejecuta instrucciones legibles con ordenador procedentes de una memoria para llevar a cabo, por medio del Cuantificador Vectorial:

- comparación (902) de un vector **s** objetivo de entrada con cuatro centroides C_0, C_1, $C_{0,flip}$ y $C_{1,flip}$, en donde el centroide $C_{0,flip}$ es una versión invertida del centroide C_0 y el centroide $C_{1,flip}$ es una versión invertida del centroide C_1, en donde los elementos de $C_{i,flip}$ se definen como $C_{i,flip} = [C_i(M), C_i(M-1), ..., C_i(1)]$, donde $C_i(m)$ son elementos de vector del centroide C_i, M es una longitud del vector, e i es un índice del centroide, representando cada centroide una clase respectiva de vectores de código y siendo obtenido como resultado de la ejecución de un algoritmo de agrupamiento;

- determinación (906) de un punto de inicio para una búsqueda relacionada con el vector objetivo de entrada en el libro de códigos, en base al resultado de la comparación, y

- realización (910) de una búsqueda en el libro de códigos, comenzando en un punto de inicio determinado, e identificando un vector de código para representar el vector **s** objetivo de entrada, en donde el número de vectores objetivo de entrada por bloque o segmento de tiempo es variable y un espacio de búsqueda se ajusta dinámicamente de tal modo que el tamaño del espacio de búsqueda depende del número de vectores objetivo de entrada, representando dichos vectores objetivo de entrada una región de pico espectral; en donde el libro de códigos ha sido creado de tal modo que los vectores de código en el libro de códigos están clasificados conforme a una medición de distorsión que refleja la distancia entre cada vector de código y dichos centroides C_0 y C_1.

2.- Método según la reivindicación 1, en donde el espacio de búsqueda se ajusta dinámicamente incrementando o reduciendo el tamaño del espacio de búsqueda en base al número de vectores objetivo.

3.- Método según la reivindicación 1 o 2, en donde los vectores de código en el libro de códigos están clasificados de tal modo que las palabras de código más próximas al primer centroide C_0 y más distanciadas de un segundo centroide C_1, están en una clase del libro de códigos, mientras que las palabras de código más próximas a C_1 y más distanciadas de C_0 están agrupadas en otra clase del libro de códigos.

4.- Códec de transformación que comprende un Cuantificador Vectorial (1000), adaptado para realizar codificación de región de pico, que comprende:

- una unidad (1002) de comunicación, adaptada para obtener un número variable de vectores **s** objetivo de entrada por bloque o segmento de tiempo;

- una unidad (1004) de comparación, adaptada para comparar un vector **s** objetivo de entrada con cuatro centroides C_0, C_1, $C_{0,flip}$, $C_{1,flip}$, en donde el centroide $C_{0,flip}$ es una versión invertida del centroide C_0 y el centroide $C_{1,flip}$ es una versión invertida del centroide C_1, en donde los elementos de $C_{i,flip}$ se definen como $C_{i,flip} = [C_i(M), C_i(M-1), ..., C_i(1)]$, donde $C_i(m)$ son elementos de vector del centroide C_i, M es la longitud del vector, e i es un índice del centroide, representando cada centroide una clase respectiva de vectores de código y siendo obtenido como resultado de la ejecución de un algoritmo de agrupamiento;

- una unidad (1008) de determinación, adaptada para determinar un punto de inicio para una búsqueda en el libro de códigos, en base al resultado de la comparación, y para ajustar dinámicamente un espacio de búsqueda de tal modo que el tamaño del espacio de búsqueda depende del número de vectores objetivo de entrada, representando dichos vectores objetivo de entrada una región de pico espectral, y

- una unidad (1010) de búsqueda, adaptada para realizar una búsqueda en el libro de códigos, comenzando en el punto de inicio determinado, e identificando un vector de código para representar el vector **s** objetivo de entrada;

en donde los vectores de código en el libro de códigos (1014) están clasificados de acuerdo con una medición de distorsión que refleja la distancia entre cada vector de código y dichos centroides C_0 y C_1.

5.- Códec de transformación según la reivindicación 4, en donde la unidad (1008) de determinación está adaptada para ajustar dinámicamente el espacio de búsqueda incrementando o reduciendo el tamaño del espacio de búsqueda en base al número de vectores objetivo.

6.- Códec de transformación según la reivindicación 4 o 5, en donde los vectores de código en el libro de códigos están clasificados de tal modo que las palabras de código más cercanas a un primer centroide C_0 y más distanciadas de un segundo centroide C_1 están en una clase del libro de códigos, mientras que las palabras de código más cercanas a C_1 y más distanciadas de C_0 están agrupadas en la otra clase del libro de códigos.

7.- Terminal móvil que comprende el códec de transformación según cualquiera de las reivindicaciones 4-6.

8.- Programa informático (1210) que comprende un código legible con ordenador, el cual, cuando se ejecuta en un procesador de un códec de transformación que comprende un Cuantificador Vectorial, hace que el Cuantificador Vectorial lleve a cabo el método correspondiente para codificación de región de pico según cualquiera de las reivindicaciones 1-3.

9.- Producto (1208) de programa informático que comprende un medio legible con ordenador y un programa informático (1210) según la reivindicación 8 almacenado en el medio legible con ordenador.

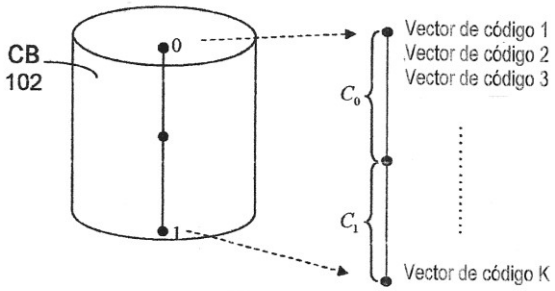

Figura 1a

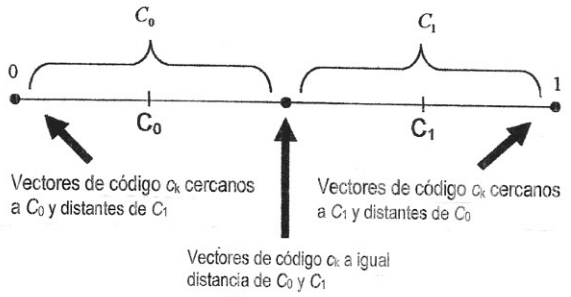

Figura 1b

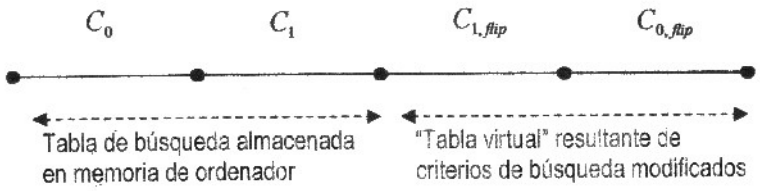

Figura 2

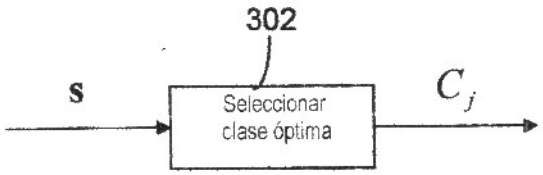

Figura 3

Figura 4

# picos	17	16	15	14	13	12	11	10	9	8	...	1
# coeficientes en la búsqueda	128	136	145	155	167	181	197	217	241	256	...	256

Figura 5

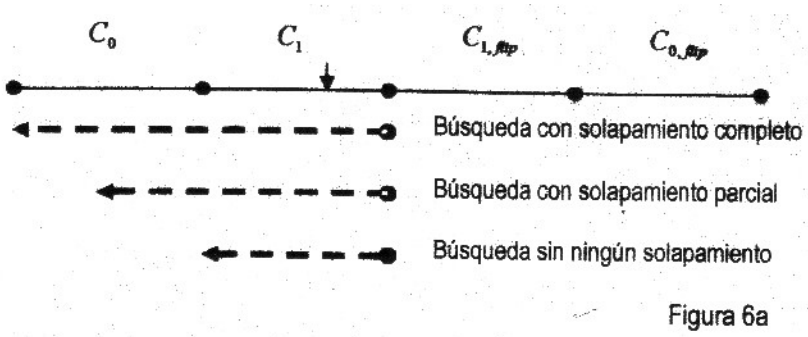

Figura 6a

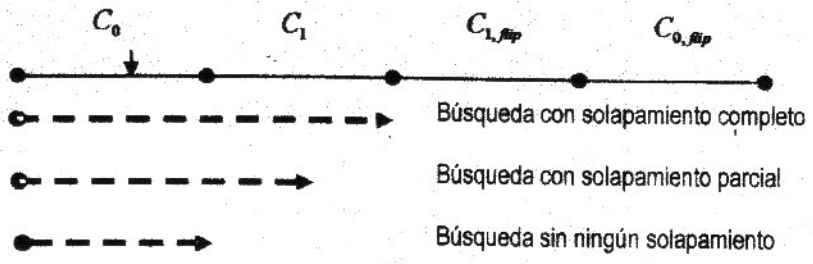

Figura 6b

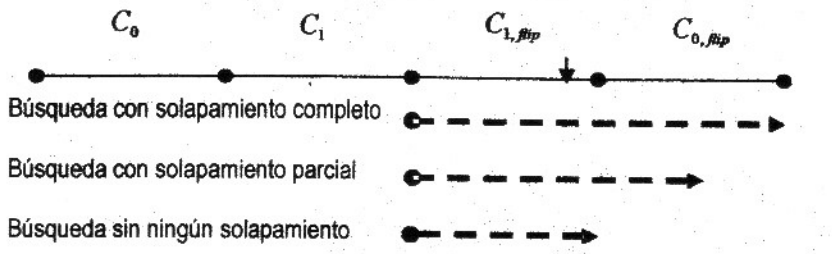

Figura 6c

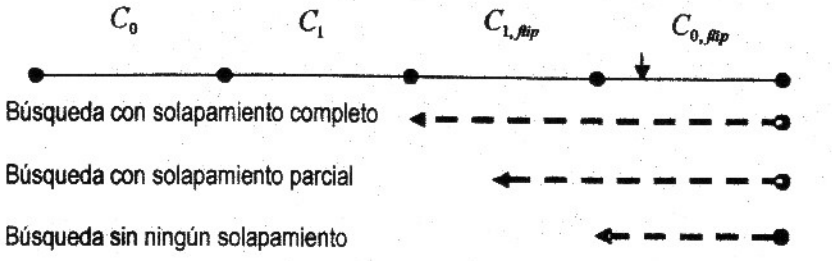

Figura 6d

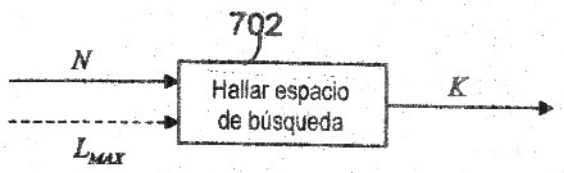

Figura 7

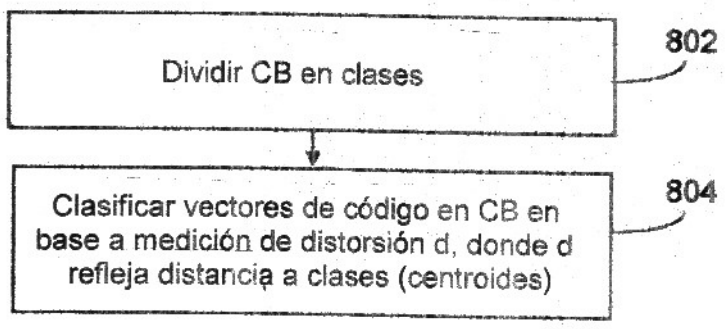

Figura 8

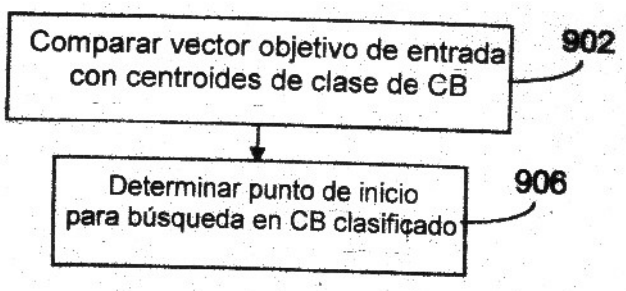

Figura 9a

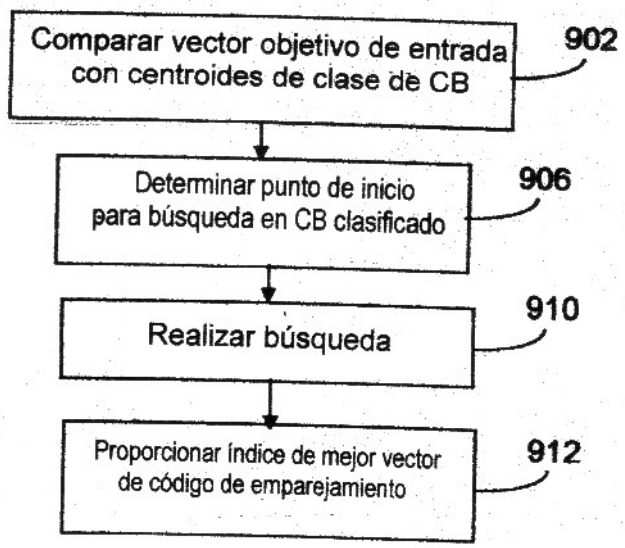

Figura 9b

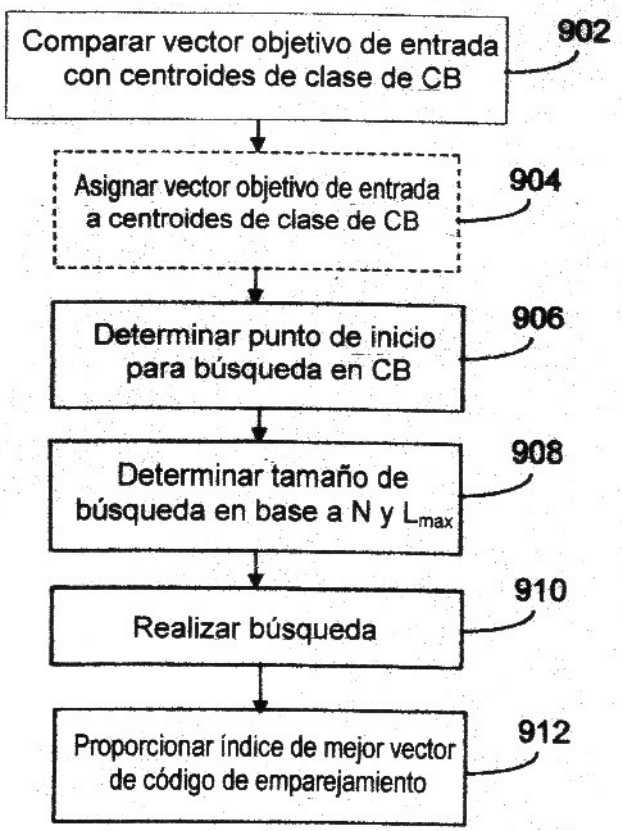
Comparar vector objetivo de entrada con centroides de clase de CB
902
Asignar vector objetivo de entrada a centroides de clase de CB
904
Determinar punto de inicio para búsqueda en CB
906
Determinar tamaño de búsqueda en base a N y L_{max}
908
Realizar búsqueda
910
Proporcionar índice de mejor vector de código de emparejamiento
912

Figura 9c

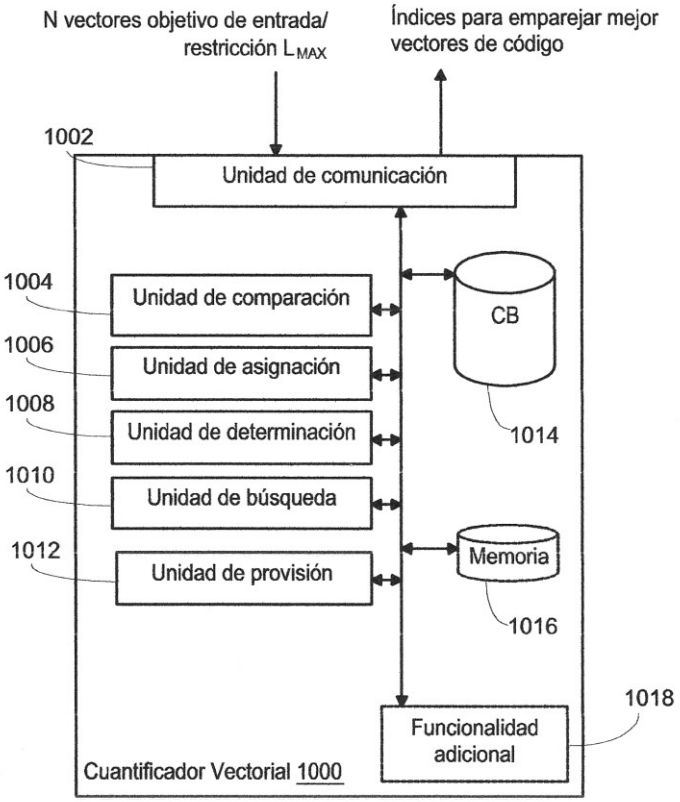
N vectores objetivo de entrada/
restricción L_{MAX}
Índices para emparejar mejor
vectores de código
1002
Unidad de comunicación
1004
Unidad de comparación
1006
Unidad de asignación
1008
Unidad de determinación
1010
Unidad de búsqueda
1012
Unidad de provisión
CB
1014
Memoria
1016
1018
Funcionalidad
adicional
Cuantificador Vectorial 1000

Figura 10

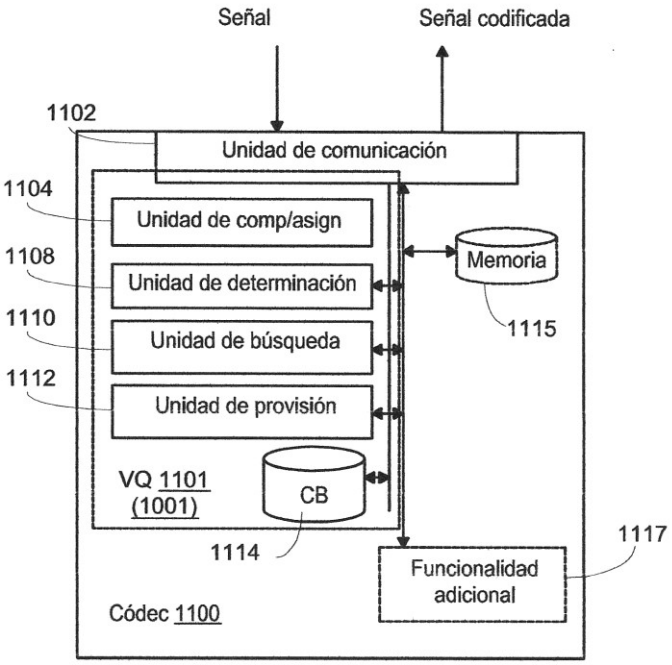

Figura 11

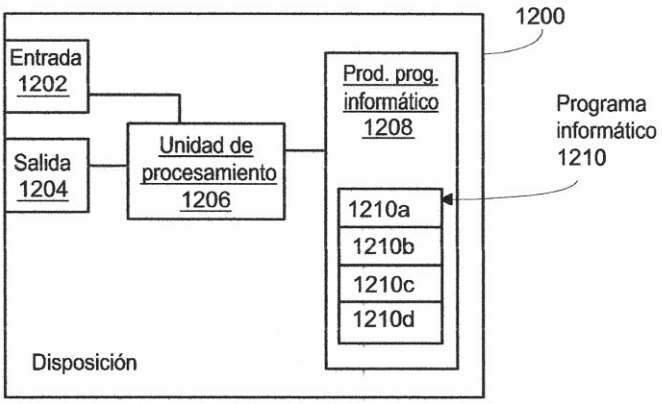

Figura 12