(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 960 582**

(51) Int. Cl.:

G06T 9/00 (2006.01)
H03M 7/30 (2006.01)
G10L 19/00 (2013.01)
G10L 19/038 (2013.01)
G10L 19/18 (2013.01)
H04N 19/94 (2014.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **12.12.2012 E 19167463 (9)**

(97) Fecha y número de publicación de la concesión europea: **09.08.2023 EP 3547261**

(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:
05.03.2024

(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:

ELZABURU, S.L.P

DESCRIPCIÓN

Cuantificador vectorial

Campo técnico

La tecnología sugerida se relaciona generalmente con la cuantificación vectorial (VQ), y especialmente con la precisión y complejidad computacional de la misma.

Antecedentes

Hay dos clases principales de algoritmos de cuantificación, a saber: cuantificadores escalares (SQ), que procesan un vector de entrada elemento por elemento, y cuantificadores vectoriales (VQ), que cuantifican un vector de entrada como una unidad (todas las dimensiones del vector se cuantifican conjuntamente). A una tasa de bits dada, los VQ son superiores a los SQ, pero a costa de una mayor complejidad computacional y almacenamiento de memoria.

Sea el vector objetivo a cuantificar M dimensional: s = [s(1) s(2)... s(M)]. El algoritmo de VQ realiza una búsqueda en un libro de códigos (CB) de tamaño k, $\{\boldsymbol{c_k}\}_{k=1}^{K}$ de vectores de código M dimensional prealmacenado C_k = [C_k (1) C_k(2)... C_k(M)]. Tal búsqueda devuelve el índice del vector de libro de códigos que proporciona la mejor coincidencia k^{opt} en base a una medida de distorsión $d(s, C_k)$. Las ecuaciones (1-2) a continuación describen esta operación, suponiendo que el criterio de búsqueda se basa en un error cuadrático:

$$k^{opt} = \underset{k}{arg\,min}\, d(\boldsymbol{s}, \boldsymbol{c}_k) \qquad (1)$$

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

El índice óptimo k^{opt} se transmite al decodificador, y el vector de código correspondiente se extrae del CB (CB idénticos están disponibles tanto en el codificador como en el decodificador) y se utiliza para reconstruir el vector objetivo. El CB típicamente se entrena 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 ponderaciones, de manera que:

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

donde las ponderaciones w(m) son dependientes de la aplicación. Para simplificar la presentación en la presente memoria, en la descripción adicional solo se utilizará el error cuadrático, definido en la ecuación (2). Sin embargo, se debería señalar que los principios discutidos en la presente memoria también son válidos cuando se usan criterios más sofisticados, tales como el de la ecuación (3).

Como se puede concluir a partir de la descripción anterior, la precisión o calidad de la señal objetivo reconstruida es dependiente del tamaño k del libro de códigos; donde un CB grande conduce a una mayor precisión y, de este modo, 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 también está relacionada con el tamaño del CB, suponiendo que la dimensionalidad del vector está fijada por la aplicación.

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

Si un sistema de VQ es para cuantificar un vector objetivo **s** a la vez, el espacio de búsqueda k tiene que ser optimizado de manera que la complejidad no exceda L_{MAX}. Algunas técnicas de optimización fuera de línea, tales como el VQ dividido y de múltiples etapas, pueden proporcionar cierta reducción de la complejidad (y el almacenamiento), dadas las propiedades del vector **s** y los requisitos de calidad para el vector reconstruido.

Si el sistema de VQ es para cuantificar múltiples vectores objetivo (de entrada) $\{\boldsymbol{s}_n\}_{n=1}^{N}$ a la vez, con un número variable de vectores N, las técnicas de optimización fuera de línea mencionadas anteriormente no son capaces de mantener las restricciones de complejidad y calidad. En tales casos, la optimización fuera de línea tiene que encontrar un equilibrio entre los requisitos contradictorios de A) limitar la complejidad (=limitar la búsqueda) cuando un gran número de vectores de entrada se van a cuantificar simultáneamente, y B) mantener una alta precisión (=buscar en un libro de códigos grande) cuando se va a cuantificar un número bajo de vectores, lo cual no es una tarea sencilla. Los artículos "Fast codebook search algorithm for unconstrained vector quantisation" de C.-Q. Chen et al y "Application of sorted codebook vector quantization to spectral coding of speech" de H.R.S. Mohammadi et al describen la técnica

anterior relevante.

Compendio

La invención está definida por las reivindicaciones adjuntas.

La tecnología descrita en la presente memoria es aplicable, por ejemplo, para sistemas de compresión/transmisión de audio y video que realizan compresión con pérdida en el flujo de entrada, y se podría describir en una serie de aspectos diferentes. La tecnología descrita en la presente memoria implica un libro de códigos, que se divide en clases y se ordena, y la clasificación de un vector 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 respecto al vector de entrada, se busque primero de las clases en el libro de códigos. Por tanto, el vector de código de mejor coincidencia para el vector de entrada se puede encontrar al principio de una búsqueda, y se puede reducir la complejidad computacional.

Según un primer aspecto, se proporciona un método de codificación de audio para codificar un número variable de vectores objetivo por segmento de tiempo. Un vector objetivo representa una región de pico espectral de un segmento de una señal de audio. El método comprende obtener un vector objetivo de entrada que comprende coeficientes de transformación en ambos lados de un pico espectral, comparando el vector objetivo de entrada con una pluralidad de centroides, es decir, vectores de referencia, donde cada centroide representa una clase respectiva de vectores de código en un libro de códigos. El método comprende además determinar un punto de inicio, en un libro de códigos, donde el punto de inicio se determina en base al resultado de la comparación, y realizar una búsqueda en el libro de códigos, comenzando en el punto de inicio determinado, para identificar un vector de código que mejor coincida con el vector objetivo de entrada. El espacio de búsqueda se ajusta dinámicamente de manera que el tamaño del espacio de búsqueda depende del número de vectores objetivo de entrada. Los vectores de código en el libro de códigos se clasifican según una medida de distorsión que refleja la distancia entre cada vector de código y los centroides. El método 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 respecto al vector de entrada, se busque primero de las clases en el libro de códigos.

Según un segundo aspecto, se proporciona un códec de audio adaptado para realizar la codificación de un número variable de vectores objetivo por segmento de tiempo, que comprende medios para ejecutar el método según el primer aspecto. El códec de audio comprende medios para obtener un vector objetivo de entrada que comprende coeficientes de transformación en ambos lados de un pico espectral y medios para comparar el vector objetivo de entrada con una pluralidad de centroides, cada centroide que representa una clase respectiva de vectores de código en un libro de códigos. El códec de audio comprende además medios 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 manera que el tamaño del espacio de búsqueda dependa del número de vectores objetivo de entrada. Los vectores de código en el libro de códigos se clasifican según una medida de distorsión que refleja la distancia entre cada vector de código y los centroides. El códec de audio permite que la clase de vectores de código que comprende los vectores de código candidatos más probables con respecto al vector de entrada se busquen primero de las clases en el libro de códigos.

Según un tercer aspecto, se proporciona un terminal móvil que comprende un códec de audio según el segundo aspecto anterior.

El tamaño de una región de búsqueda en el libro de códigos, en el que se realiza una búsqueda, se puede adaptar en base a un número de vectores objetivo de entrada y una restricción de complejidad máxima. El número de vectores objetivo de entrada por unidad de codificación puede ser variable. Además, la restricción de máxima complejidad se puede establecer dinámicamente. Además, se podría realizar una búsqueda en el libro de códigos en el espacio de búsqueda determinado, comenzando en el punto de inicio determinado, donde la búsqueda entrega la mejor coincidencia con el vector objetivo de entrada. Por "mejor coincidencia" se entiende aquí la coincidencia más cercana, la distancia más corta, con respecto a una medida de distancia entre el vector objetivo de entrada y un vector candidato en el libro de códigos, es decir, una mejor coincidencia es un vector de código que tiene la distancia más corta al vector objetivo de entrada, según la medida de la distancia.

Breve descripción de los dibujos

La tecnología sugerida se describirá ahora con más detalle por medio de realizaciones ejemplares y con referencia a los dibujos adjuntos, en los que:

La figura 1a y 1b muestra la estructura de un CB ordenado según una solución descrita en la presente memoria. La búsqueda comienza desde el punto 0 o punto 1 hacia el otro extremo del CB.

La figura 2 muestra un ejemplo de estructura de CB que explota la simetría. Solo los vectores de código de C_0 y C_1 se almacenan en una memoria ROM.

La figura 3 ilustra una unidad funcional ejemplar para determinar una clase óptima para un vector de entrada **s** comparando el vector de entrada **s** con cada uno de un número de centroides $\{C_0\ C_1\ C_{\bar{1},voltear}\ C_{0,voltear}\}$, cada uno asociado con una clase en un libro de códigos.

La figura 4 ilustra una unidad funcional para determinar el 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 la tasa de bits del códec.

La figura 5 es una tabla que ilustra que una región de búsqueda aumenta a medida que disminuye el número de picos por trama. En el ejemplo; con 17 picos (=17 vectores de entrada) la búsqueda se realiza solo en el CB de 7 bits (definido que es el espacio de búsqueda mínimo en este ejemplo), pero con 8 picos o menos, la búsqueda se realiza en el CB de 8 bits (espacio de búsqueda máximo), dado que esto se puede "permitir" bajo la restricción de máxima complejidad.

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

La figura 7 ilustra que la complejidad permitida L_{MAX} se puede señalar al sistema desde una entidad externa. El parámetro L_{MAX} se podría basar, por ejemplo, en la carga de la CPU o estado de la batería.

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

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

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

La figura 12 es un diagrama de bloques que ilustra una disposición para la cuantificación de vectores, según un ejemplo de la tecnología sugerida en la presente memoria.

Descripción detallada

Brevemente descrita, la solución descrita en la presente memoria se relaciona con la adaptación dinámica del espacio de búsqueda de un VQ, de manera 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 calidad dentro de una restricción de complejidad dada. Es decir, los requisitos de complejidad computacional (véase L_{MAX}) no van a ser violados. Esto se logra porque 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 aumenta o reduce, en base al número de vectores objetivo. El algoritmo de VQ descrito en la presente memoria se puede considerar como una "herramienta" para la compresión de datos, independientemente de cuáles sean los datos, es decir, los datos podrían ser, por ejemplo, vídeo 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 códecs de audio. Por ejemplo, también se podría implementar en códecs de video.

El algoritmo descrito en la presente memoria se basa en un CB especialmente diseñado. Algunas variantes de dicho libro de códigos se describirán con más detalle a continuación. Primero se describirá un caso básico, y más adelante se discutirá un esquema más avanzado. Los vectores de código del CB se pueden disponer según la solución descrita en la presente memoria en un modo fuera de línea.

Con el fin de crear una versión básica del CB ventajoso especialmente diseñado, los vectores de código de un CB se dividen en dos clases, aquí indicadas C_0 y C_1 (esta notación se utilizará tanto para los nombres de las clases, como para los correspondientes centroides, véase la figura 1). Para dividir los datos en dos clases, se puede utilizar el denominado algoritmo K-medias (algoritmo de Lloyd generalizado). Es una técnica bien conocida, que toma un conjunto de datos completo como entrada, y el número deseado de clases, y emite los centroides del número deseado de clases. Por ejemplo, el algoritmo emite 2 vectores centroides si el número deseado de clases se ha indicado en 2. Tenga en cuenta que estos centroides, cuando se usa un algoritmo de K-medias, son vectores de la misma dimensión que los vectores del conjunto de datos, pero no pertenecen al conjunto de datos. Es decir, los vectores centroides están fuera del CB y no necesitan coincidir con algunos vectores de código existentes. Por "centroide" se entiende en la presente memoria generalmente un vector de referencia que representa una clase de vectores.

Todos los vectores de código en el CB se clasifican luego según una medida de distorsión, por ejemplo, como la definida en la ecuación (4)

$$d(\boldsymbol{c}_k, C_0, C_1) = \sum_{m=1}^{M}(c_k(m) - C_0(m))^2 - \sum_{m=1}^{M}(c_k(m) - C_1(m))^2 \qquad (4)$$

La medida de distorsión anterior da como resultado, o supone, valores negativos grandes para vectores de código cercanos a C_0, y valores positivos grandes para vectores de código cercanos a C_1. Los vectores de código que están igualmente distanciados de los centroides (C_0 y C_1) de las dos clases producen una medida de distorsión d que es

cercana a cero. En el CB, los vectores de código están ordenados, por ejemplo, aumentando la medida de distorsión, como se ilustra en la figura 1a y 1b.

Cada vector de destino de entrada se compara con los dos centroides (el centroide respectivo de las dos clases) y, dependiendo del resultado, se asigna, es decir, se concluye o se determina que pertenece a cualquiera de la clase C_0 o la clase C_1. En base a esa clasificación, el punto de inicio de la búsqueda se selecciona para que sea o bien el punto más alto (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 bajo (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 se debería hacer dependiente del número de vectores objetivo de entrada N por bloque o segmento/intervalo de tiempo. Si redefinimos que el espacio de búsqueda K no sea del tamaño de todo el CB, sino que sea variable, el concepto detrás de la adaptación descrita en la presente memoria se puede definir en la ecuación (5)

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

En otras palabras *#Cuantificadores* × *#Operaciones_por_cuantificador* ≈ *const,* donde "Cuantificador" se puede considerar como el algoritmo que mapea un vector de entrada a uno de los vectores de código.

En la presente memoria, como ejemplo, el VQ se describe en el contexto de un códec de transformación que codifica picos espectrales, o estrictamente, las regiones alrededor de los picos espectrales. En el contexto de tal códec, un vector objetivo de entrada puede reflejar un pico espectral (región) de un segmento de la señal de audio que se está procesando. El número de picos espectrales en el espectro de 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. Dado 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. Por tanto, cuando se usa un codificador de transformación que codifica regiones de pico espectral, el número de vectores de entrada, por bloque o segmento de tiempo, variará para el VQ. En los ejemplos en 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 solo un ejemplo y no se debería interpretar como una limitación de la solución en general.

Efectivamente, el esquema descrito anteriormente mantiene el número de operaciones requeridas para el VQ en un rango estrecho (o casi constante); es decir, cuando aumenta el número de VQ, es decir, aumenta el número de vectores objetivo de entrada, disminuye el número de operaciones por VQ (el tamaño del espacio de búsqueda disminuye/solo se busca una parte del CB), de manera que los requisitos de complejidad (es decir, restricciones) no se violan. Con una disminución de N, el espacio de búsqueda k se puede aumentar, como máximo, hasta el tamaño de todo el CB, lo que conduce a una mayor precisión y, por tanto, calidad del vector reconstruido. La precisión de un cuantificador vectorial se puede medir como un error cuadrático entre una señal original y los datos reconstruidos correspondientes.

De esta forma, no necesita ser diseñado el libro de códigos del VQ para el peor de los casos (es decir, el número máximo de vectores objetivo de entrada). En su lugar, se podría diseñar, por ejemplo, en el mejor de los casos, comprendiendo por tanto más vectores de código de los que se podrían buscar para el número máximo de vectores objetivo de entrada dentro de la restricción de máxima complejidad L_{MAX}.. El requisito de máxima complejidad se cumplirá si la extensión de la búsqueda, es decir, el tamaño del espacio de búsqueda, en el CB depende 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 sufriría enormemente, dado que no habría forma de saber dónde se sitúa el vector de "mejor coincidencia" en el CB, o si este vector de mejor coincidencia se sitúa en una parte del libro de códigos que se buscará 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. Se debería señalar que el diseño de CB descrito en la presente memoria también es beneficioso para aplicaciones en las que el número de vectores de entrada, por unidad de codificación, es constante.

Ejemplo de realización 1: VQ restringido en regiones de picos espectrales

Un conjunto de vectores objetivo s representan regiones de picos espectrales en una codificación de audio de dominio de transformación, por ejemplo, coeficientes de transformación en la vecindad de los picos de MDCT. Por tanto, 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 objetivo s exhiben ciertas simetrías que se pueden utilizar para optimizar aún más el CB. Por ejemplo, los coeficientes de transformación en ambos lados de un pico espectral tienen estadísticas similares. Si asumimos que los vectores objetivo s están centrados en la posición pico, la simetría descrita anteriormente permite añadir una estructura adicional en el CB ordenado de la figura 1a y 1b. La estructura de tal CB nuevo y mejorado además se ilustra en la figura 2. La figura 2 ilustra un CB en el que los vectores de código de la parte izquierda se almacenan en una memoria, tal como una memoria de solo lectura. Sin embargo, no hay vectores de código almacenados previamente para el lado derecho del CB, es decir, para las

clases $C_{1,voltear}$ y $C_{0,voltear}$. Los vectores de código de estas clases son versiones volteadas de los vectores de código del lado derecho del CB. Por tanto, cuando se realiza una búsqueda en las clases $C_{0,voltear}$ y $C_{1,voltear}$ la búsqueda se realiza en los vectores de código del lado izquierdo, que se almacenan en la memoria, pero con los elementos de los vectores de código volteados alrededor del centro, de manera que los vectores de código $\mathbf{C}_{k,\ voltear}$ están dados por la ecuación (6)

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

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 cierto vector de código en C_0 son {C_{01} C_{02} C_{03} C_{04}}, los elementos de un vector de código correspondiente en $C_{0,voltear}$ son {C_{04} C_{03} C_{02} C_{01}}

Cuando se usa un CB como el ilustrado en la figura 2, un vector objetivo de entrada se compara con cuatro centroides y se asigna a una clase con el fin de determinar un punto de inicio para la búsqueda, es decir, la clase óptima para el vector objetivo de entrada se determina comparando el vector de entrada s con cada uno de los centroides {C_0 C_1 $C_{1,voltear}$ $C_{0,voltear}$}. Esto se ilustra en la figura 3, donde un vector objetivo s se introduce a una unidad de asignación de clase 302, que entrega un indicador de clase, C_j, como salida. Los centroides $C_{1,voltear}$ y $C_{0,voltear}$ no se almacenan en una tabla, sino que se "crean" volteando los elementos de los centroides C_0 y C_1. Los elementos no necesitan ser volteados literalmente; en su lugar, una operación de búsqueda modificada puede leer los elementos de C_0 y C_1 en el orden inverso cuando se leen $C_{0,voltear}$ y $C_{1,voltear}$, es decir, tanto centroides como vectores de libro de códigos. De esta forma, el CB se puede extender para comprender dos veces el número de vectores de código, en comparación con lo que se almacena físicamente en el CB, lo que significa un ahorro de recursos de memoria. Esto es posible debido a que se explota la simetría de las regiones de picos, como se describió anteriormente. Más específicamente, la solución se basa en la observación de que la simetría se puede explotar si un vector de código válido invertido también es un vector de código válido.

La región de búsqueda se adapta al número de picos espectrales, que corresponde al número de vectores objetivo de entrada. Esto se ejemplifica en la figura 4, que muestra una unidad funcional 402 que toma un indicador de un número de picos/vectores como entrada, y que produce un indicador de una región de búsqueda como salida. La figura 4 ilustra además que, por ejemplo, la tasa de bits de un códec que aplica el VQ se podría tener en cuenta cuando se determina la región de búsqueda (espacio de búsqueda). Por ejemplo, puede haber diferentes requisitos de calidad para diferentes tasas de bits, por ejemplo, cuanto mayor sea la tasa de bits, mayor será la calidad esperada. Además, la complejidad máxima permitida puede cambiar con la tasa de bits, por ejemplo, dado que a diferentes velocidades de bits se activan diferentes módulos en un códec, que no son igualmente complejos, es decir, la complejidad permitida restante para el VQ, a partir de una restricción de complejidad máxima en todo el procedimiento de codificación, podría no ser la misma. Es decir, la información de tasa de bits refleja cambios en los requisitos de calidad y complejidad, que se pueden tener en cuenta en la cuantificación del 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 se indica como el número de coeficientes (vectores de código) en la búsqueda por vector de entrada. Los números en la tabla de la figura 5 se derivan 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 cuales dos son "normales" o "físicos" y dos están "volteados" o son "virtuales").

La lógica detrás de la tabla en la figura 5 es que cuando hay un pico menos a codificar, por ejemplo, 16 en lugar de 17, las 128 comparaciones "ahorradas" se pueden distribuir entre los picos restantes. En algún momento, la longitud de búsqueda se satura, porque 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 hacer una búsqueda completa de todos los vectores objetivo de entrada (es decir, picos) sin llegar a la complejidad máxima permitida.

Los ejemplos del procedimiento de búsqueda se ilustran en la figura 6a-d. En la práctica, es el mismo tipo de búsqueda que la descrita anteriormente junto con la figura 1a-b, pero con una clasificación adicional en segmentos/clases de CB "normales" y "volteados".

En el ejemplo mostrado en la figura 6a, el vector de entrada s pertenece a la clase C_1 (posición indicada con la flecha hacia abajo). El espacio de búsqueda se limita entonces a tamaños entre la clase C_1 solamente, y el espacio conjunto de las clases C_0 y C_1. En la figura 6a, esto se ilustra con tres flechas discontinuas que indican espacios de búsqueda de diferentes tamaños. La figura 6b ilustra el caso donde un vector de entrada pertenece a la clase C_0 (posición indicada con flecha hacia abajo), en cuyo caso la búsqueda tiene un punto de inicio diferente.

Análogamente, como se ilustra en las figuras 6c-d, la búsqueda se puede realizar en clases $C_{1,voltear}$ y/o $C_{0,voltear}$ si el vector de entrada pertenece a una de estas clases. No se realizan búsquedas en el espacio conjunto de C_1 y $C_{1,voltear}$. La razón de esto es que el espacio conjunto de una clase normal y una volteada no corresponde a un conjunto de datos reales (sus estadísticas no corresponden a estadísticas de datos reales). Por tanto, no es muy probable que se pueda encontrar una buena coincidencia para un vector de entrada en tal espacio.

Realización ejemplar 2: sistema de comunicación con control externo de máxima complejidad permitida

El concepto de un VQ que tiene una complejidad que se ajusta dinámicamente al número de vectores objetivo *N* se puede extender al caso cuando el límite de complejidad no está predeterminado, pero puede variar, por ejemplo, en base a algún criterio, y se puede señalar al VQ y/o a la entidad en la que se aplica el VQ. Esto se ilustra en el diagrama de bloques esquemático de la figura 7, que muestra una unidad funcional 702 que toma un indicador o el número de picos/vectores como entrada y, además, que toma una restricción de complejidad L_{MAX} como entrada. La unidad funcional 702 entrega un indicador de un espacio de búsqueda/una región del CB, véanse las flechas discontinuas en las figuras 6a-d.

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

Procedimiento ejemplar para lograr una estructura de CB

A continuación, con referencia a la figura 8, se describirá un procedimiento ejemplar para diseñar u organizar un CB para uso en un VQ. El procedimiento es para crear un CB para uso en un VQ que proporciona cuantificación en codificador de audio transformado, tal como, por ejemplo, un codificador de MDCT.

El procedimiento descrito a continuación se relaciona con las partes de un procedimiento de creación de CB que se desvían y/o son adicionales a la creación u organización de un CB de VQ convencional.

El CB se divide en clases en una acción 802, por ejemplo, mediante el uso de un denominado algoritmo de K-medios, como se describió anteriormente. Los vectores de código del CB se ordenan luego en el CB en base a una medida de distorsión, por ejemplo, como la descrita en la ecuación (4). La medida de distorsión para cada vector de código depende de una relación entre el vector de código y los centroides que representan cada clase de CB, como se describió anteriormente.

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

Procedimiento de VQ ejemplar

A continuación se describirá un procedimiento ejemplar en un cuantificador vectorial (VQ), con referencia a la figura 9a. El procedimiento es adecuado para uso en un codificador de audio transformado, 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 se reciben por el VQ, como se describió anteriormente. A continuación, se describirán las acciones asociadas con uno de los vectores objetivo de entrada, por razones de simplicidad.

Un vector objetivo de entrada s se compara con una serie de vectores de código, cada uno que representa una clase de CB (véanse las clases C_0 y C_1, etc. descritas anteriormente), preferiblemente el centroide de cada clase. La comparación se ilustra como acción 902 en la figura 9a-c. La acción 902 se podría considerar alternativamente como integrada con una acción 904, ilustrada en la figura 9c. Dependiendo del resultado de las comparaciones, al vector objetivo de entrada s se le asigna una de las clases, o secciones, del CB, en una acción 904. Al vector objetivo de entrada s se le asigna, o se concluye que pertenece a, la clase a la que tiene la distancia más corta, es decir, a la que es más similar, según alguna medida de distancia (medida 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 medida 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 sobre 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 ventajosamente diseñado y organizado, la probabilidad de que la mejor coincidencia, de todos los vectores de código candidatos dentro de todo el CB, para el vector objetivo de entrada s se encuentre dentro del espacio de búsqueda es muy alta, incluso cuando el espacio de búsqueda está limitado A, por ejemplo, la mitad del CB. En un caso en el que el espacio de búsqueda comprendiera todo el libro de códigos, el vector de código de mejor coincidencia se encontraría pronto en la búsqueda cuando se inicia la búsqueda en el punto de inicio determinado.

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

Además, el tamaño del espacio de búsqueda se puede determinar en una acción 908 ilustrada en la figura 9c. El espacio de búsqueda se puede describir como el número de vectores de código en el CB que se deberían evaluar en la búsqueda de la mejor coincidencia para el vector objetivo de entrada s. El tamaño del espacio de búsqueda se determina en base al número de vectores objetivo de entrada y una restricción sobre la complejidad computacional

L_{MAX}. Por tanto, el tamaño del espacio de búsqueda se puede determinar, por ejemplo, una vez por cada bloque de codificación, o algún otro intervalo de tiempo, dependiendo, por ejemplo, de las características de la señal a ser cuantificada y/o los atributos de un códec. Si el número de vectores objetivo de entrada y la restricción L_{MAX} son constantes o semiconstantes con el tiempo, el tamaño del espacio de búsqueda también se puede mantener constante durante el tiempo correspondiente.

Luego, disposición de VQ ejemplar

A continuación, se describirá con referencia a la figura 10 una disposición de VQ ejemplar adecuada para uso en un codificador/códec de transformación. El códec de transformación podría ser, por ejemplo, un códec de MDCT. El VQ está adaptado para realizar las acciones del procedimiento descrito anteriormente.

El VQ 1001 se ilustra para comunicarse con otras entidades (por ejemplo, un códec de audio) a través de una unidad de comunicación 1002. El VQ puede comprender además otras unidades funcionales 1016, tales como, por ejemplo, unidades funcionales que proporcionan funciones regulares, y pueden comprender además una o más unidades de almacenamiento 1014.

El VQ 1001 se podría implementar, por ejemplo, por uno o más de: un procesador o un microprocesador y software adecuado con almacenamiento adecuado, por lo tanto, un Dispositivo Lógico Programable (PLD) u otro componente o componentes y/o circuitos electrónicos.

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

El VQ puede comprender una unidad de comparación 1004, que está adaptada para comparar un vector objetivo de entrada s con vectores que representan cada clase del CB, por ejemplo, el vector centroide de cada clase. Además, el VQ puede comprender una unidad de asignación 1006, que está adaptada para asignar una clase al vector objetivo de entrada s (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 de determinación 1008, 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 se puede adaptar además 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 una restricción de complejidad computacional.

Además, el VQ puede comprender una unidad de búsqueda 1010, que está adaptada para realizar una búsqueda en el CB, comenzando en el punto de inicio determinado y buscando en el espacio de búsqueda determinado. La búsqueda debe dar como resultado uno o más índices de CB que apunten al vector de código que mejor coincida con el vector objetivo de entrada s. El VQ puede comprender además una unidad de suministro 1012, que está adaptada para proporcionar dicho índice o índices a otra entidad, por ejemplo, a (o para uso por) un códec de transformación.

Disposición ejemplar

La figura 12 muestra esquemáticamente una realización de una disposición 1200 adecuada para uso, por ejemplo, en un decodificador de audio transformado, que también puede ser una forma alternativa de describir una realización del VQ ilustrado en la figura 5. Comprendida en la disposición 1200 están aquí una unidad de procesamiento 1206, por ejemplo, con un DSP (Procesador de Señal Digital). La unidad de procesamiento 1206 puede ser una sola unidad o una pluralidad de unidades para realizar diferentes pasos de los procedimientos descritos en la presente memoria. La disposición 1200 también puede comprender la unidad de entrada 1202 para recibir señales, tales como los vectores objetivo de entrada e indicadores de, por ejemplo, la tasa de bits y/o restricción de complejidad; y además la unidad de salida 1204 para señal o señales de salida, tales como los índices de CB para los vectores de código de mejor coincidencia. La unidad de entrada 1202 y la unidad de salida 1204 se pueden disponer como una sola en el hardware de la disposición.

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

Por lo tanto, en las realizaciones ejemplares descritas, los medios de código en el programa informático 1210 de la disposición 1200 pueden comprender un módulo de comparación 1210a para comparar un vector objetivo de entrada con centroides de clase de un CB. El programa informático puede comprender un módulo de asignación 1210b para asignar una clase al vector objetivo de entrada. El programa informático 1210 puede comprender además una unidad de determinación 1210c 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 de búsqueda 121 0d para buscar el CB según lo anterior. Además, el programa informático 1210 puede comprender un módulo de suministro 1210e, para proporcionar índices, que se emiten desde la búsqueda a otras entidades.

El programa informático 1210 tiene la forma de un código de programa informático estructurado en módulos de programas informáticos. 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 los diferentes módulos 1210a-d se ejecutan en la unidad de procesamiento 1206, corresponden al menos a las unidades 1004-1012 de la figura 10.

Aunque los medios de código en la realización descrita anteriormente en conjunto con la figura 12 se implementan como módulos de programas informáticos que, cuando se ejecutan en la unidad de procesamiento, hacen que la disposición y/o el codificador de audio transformado realice los pasos descritos anteriormente en conjunto con las figuras mencionadas anteriormente, en al menos uno de los medios de código se puede implementar en realizaciones alternativas al menos parcialmente como circuitos de hardware.

Si bien la tecnología sugerida se ha descrito con referencia a realizaciones de ejemplo específicas, la descripción en general solo se pretende que ilustre el concepto y no se debería considerar como limitante del alcance de la tecnología descrita en la presente memoria. Las diferentes características de las realizaciones ejemplares anteriores se pueden combinar de diferentes formas según la necesidad, los requisitos o la preferencia.

La solución descrita anteriormente se puede utilizar siempre que se apliquen los VQ, por ejemplo, en códecs en dispositivos tales como terminales móviles, tabletas, ordenadores, teléfonos inteligentes, etc.

Se ha de entender que la elección de unidades o módulos que interactúan, así como la denominación de las unidades, son solo con fines ejemplares, y los nodos adecuados para ejecutar cualquiera de los métodos descritos anteriormente se pueden configurar en una pluralidad de formas alternativas con el fin de ser capaces de ejecutar las acciones de proceso sugeridas.

Las funciones de los diversos elementos, incluyendo los bloques funcionales, incluyendo, pero no limitado a, los etiquetados o descritos como "unidad funcional", "procesador" o "controlador", se pueden proporcionar a través del uso de hardware, como hardware de circuito y/o hardware capaz de ejecutar software en forma de instrucciones codificadas almacenadas en un medio legible por ordenador. Por tanto, tales funciones y bloques funcionales ilustrados se han de entender como que están implementados en hardware y/o implementados por ordenador y, por tanto, implementados en 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, pero no limitada a, circuito o circuitos integrados de aplicaciones específicas (ASIC), y (cuando sea apropiado) máquinas de estado capaces de realizar tales funciones.

Abreviaturas

SQ	Cuantificación escalar
VQ	Cuantificación vectorial
CB	Libro de códigos
WMOPS	Millones de Operaciones por Segundo Ponderadas
MDCT	Transformada de Coseno Discreta Modificada

REIVINDICACIONES

1. Un método de codificación de audio para codificar un número variable de vectores objetivo por segmento de tiempo, un vector objetivo que representa una región de pico espectral de un segmento de una señal de audio, el método que se realiza por un códec de audio, el método que comprende:

- obtener un vector objetivo de entrada que comprende coeficientes de transformada en ambos lados de un pico espectral;

- comparar (902) el vector objetivo de entrada con cuatro centroides C_0, C_1, $C_{0,voltear}$ y $C_{1,voltear}$, en donde el centroide $C_{0,voltear}$ es una versión volteada del centroide C_0 y centroide $C_{1,voltear}$ es una versión volteada del centroide C_1, en donde los elementos del centroide $C_{i,voltear}$ se definen como $C_{i,voltear} = [C_i(M)\ C_i(M\text{-}1)...\ C_i(1)]$, en donde $C_i(m)$ son elementos vectoriales del centroide C_i, M es una longitud del vector e i es un índice del centroide, cada centroide que representa una clase respectiva de vectores de código en un libro de códigos;

- determinar (906) un punto de partida para una búsqueda basada en el resultado de la comparación;

- realizar (910) una búsqueda en el libro de códigos, comenzando en el punto de inicio determinado, para identificar un vector de código que mejor coincida con el vector objetivo de entrada, en donde un espacio de búsqueda se ajusta dinámicamente de manera que el tamaño del espacio de búsqueda depende del número de vectores objetivo de entrada, y en donde el libro de códigos se ha creado de manera que los vectores de código en el libro de códigos se ordenen según una medida de distorsión que refleja la distancia entre cada vector de códigos y dichos centroides C_0 y C_1; y

- proporcionar (912) un índice del vector de código para la transmisión a un decodificador.

2. El método según la reivindicación 1, en donde el espacio de búsqueda se ajusta dinámicamente aumentando o reduciendo el tamaño del espacio de búsqueda en base al número de vectores objetivo de entrada.

3. El método según la reivindicación 1 o 2, en donde los vectores de código en el libro de códigos se ordenan de manera que las palabras de código más cercanas a un primer centroide C_0 y más distanciadas a un segundo centroide C_1 estén en una clase del libro de códigos, mientras que las palabras de código más cercanas a C_1 y más distanciadas de C_0 se agrupan en la otra clase del libro de códigos.

4. El método según cualquiera de las reivindicaciones anteriores, en donde los vectores de código para la clase $C_{0,voltear}$ son versiones volteadas de vectores de código para la clase C_0 y vectores de código para la clase $C1_{,voltear}$ son versiones volteadas de vectores de código para la clase C_1.

5. El método según la reivindicación 4, en donde los elementos de los vectores de código $c_{k,voltear}$ en la clase $C_{j,voltear}$ se definen como:

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

donde $c_k(m)$ son elementos vectoriales del vector de código en la clase C_j y M es una longitud del vector de código y k es un índice del vector de código.

6. Un códec de audio (1100) adaptado para realizar la codificación de un número variable de vectores objetivo por segmento de tiempo, un vector objetivo que representa una región de pico espectral de un segmento de una señal de audio, el códec de audio que comprende:

- medios (1002, 1102) para obtener un vector objetivo de entrada que comprende coeficientes de transformada a ambos lados de un pico espectral;

- medios (1004, 1104) para comparar el vector objetivo de entrada con cuatro centroides C_0, C_1, $C_{0,voltear}$ y $C_{1,voltear}$, en donde el centroide $C_{0,voltear}$ es una versión volteada del centroide C_0 y centroide $C_{1,voltear}$ es una versión volteada del centroide C_1, en donde los elementos del centroide $C_{i,voltear}$ se definen como $C_{i,voltear} = [C_i(M)\ C_i(M\text{-}1)...C_i(1)]$, donde $C_i(m)$ son elementos vectoriales del centroide C_i M es una longitud del vector e i es un índice del centroide, cada centroide que representa una clase respectiva de vectores de código en un libro de códigos;

- medios (1008, 1108) 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 manera que el tamaño del espacio de búsqueda dependa del número de vectores objetivo de entrada;

- medios (1010, 1110), para realizar una búsqueda en el libro de códigos, comenzando en el punto de inicio determinado, para identificar un vector de código que mejor coincida con el vector objetivo de entrada, y en donde los vectores de código en el libro de códigos (1014, 1114) se clasifican según a una medida de distorsión que refleja la distancia entre cada vector de código y dichos centroides C_0 y C_1; y

- medios (1012, 1112) para proporcionar un índice del vector de código para transmisión a un decodificador.

7. El códec de audio según la reivindicación 6, en donde los medios (1008, 1108) para ajustar el espacio de búsqueda están configurados para ajustar dinámicamente el espacio de búsqueda aumentando o reduciendo el tamaño del espacio de búsqueda en base al número de vectores objetivo de entrada.

8. El códec de audio según la reivindicación 6 o 7, en donde los vectores de código en el libro de códigos se clasifican de tal manera que las palabras de código más cercanas a un primer centroide C_0 y más distanciadas a 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 se agrupan en la otra clase del libro de códigos.

9. El códec de audio según cualquiera de las reivindicaciones 6 a 8, en donde los vectores de código para la clase $C_{0,voltear}$ son versiones volteadas de vectores de código para la clase C_0 y vectores de código para la clase $C1_{,voltear}$ son versiones volteadas de vectores de código para la clase C_1.

10. El códec de audio según la reivindicación 9, en donde los elementos de los vectores de código $c_{k,voltear}$ en la clase $C_{j,voltear}$ se definen como:

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

donde $c_k(m)$ son elementos vectoriales del vector de código en la clase C_j y M es una longitud del vector de código y k es un índice del vector de código

11. Un terminal móvil que comprende el códec de audio según cualquiera de las reivindicaciones 6 a 10.

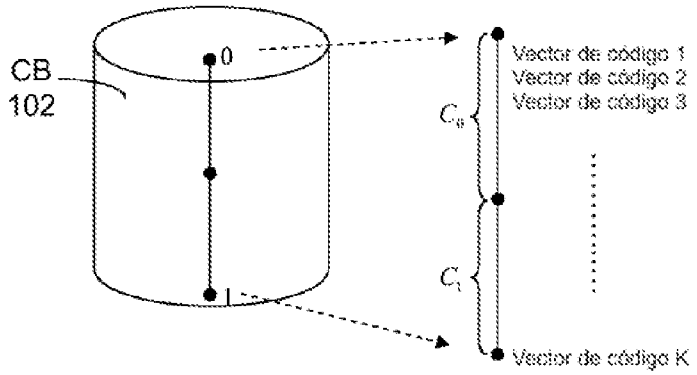

Figura 1a

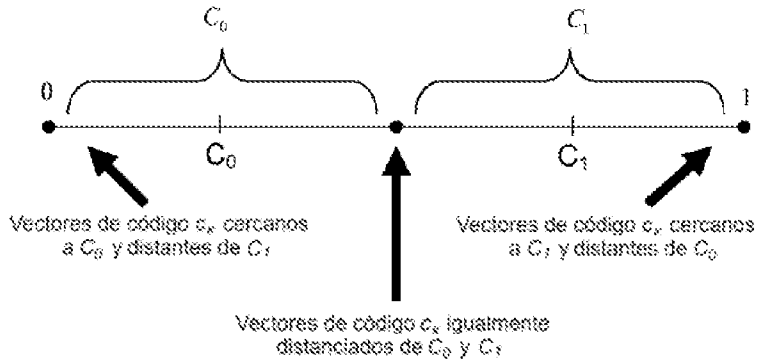

Figura 1b

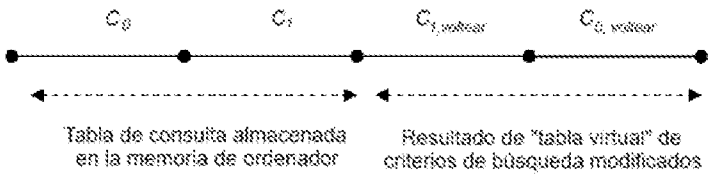

Figura 2

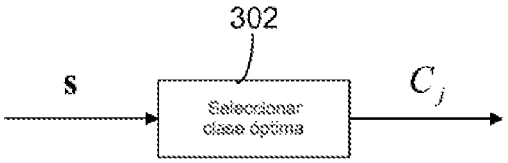

Figura 3

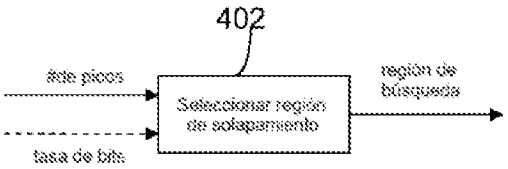

Figura 4

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

Figura 5

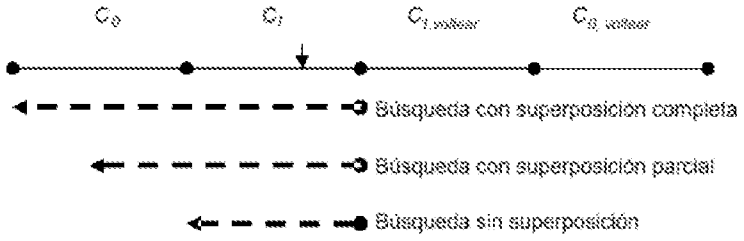

Figura 6a

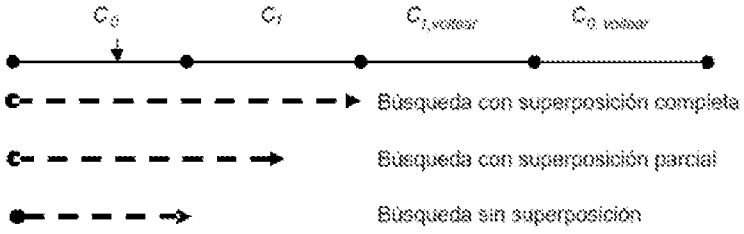

Figura 6b

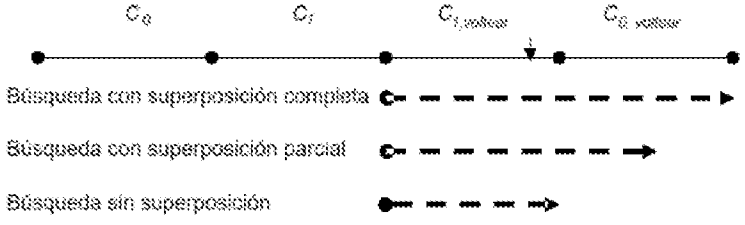

Figura 6c

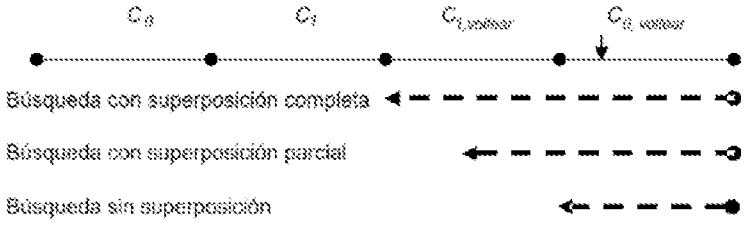

Figura 6d

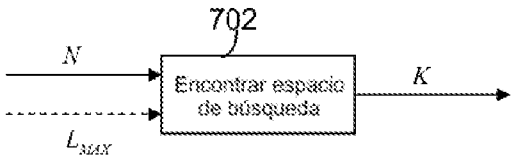

Figura 7

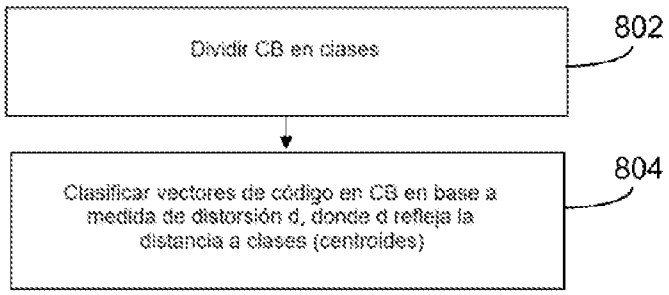

Figura 8

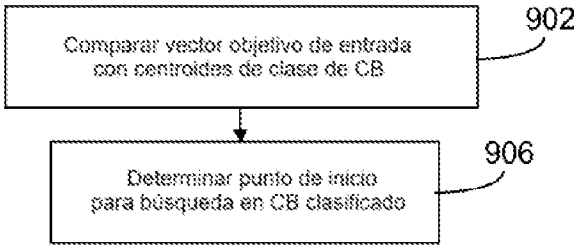

Figura 9a

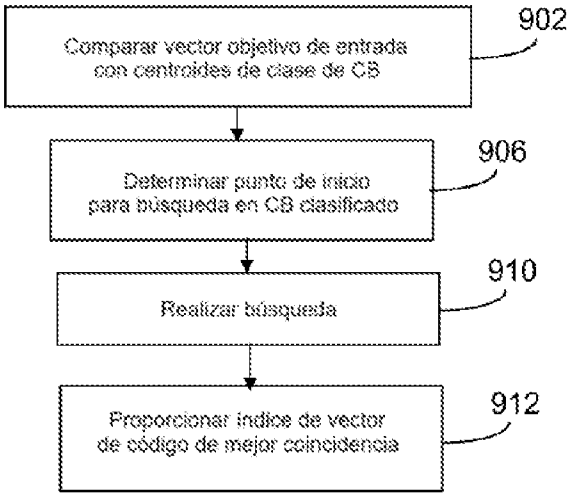

Figura 9b

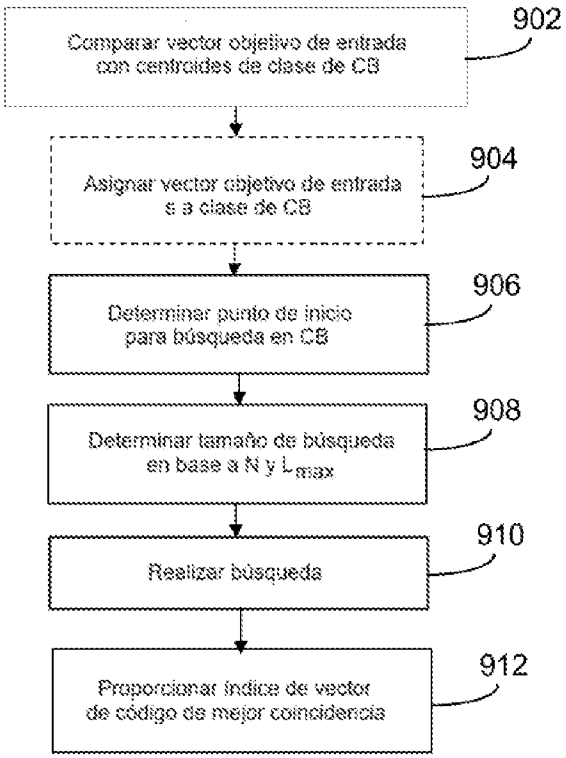
Comparar vector objetivo de entrada con centroides de clase de CB
902
Asignar vector objetivo de entrada s a clase de CB
904
Determinar punto de inicio para búsqueda en CB
906
Determinar tamaño de búsqueda en base a N y L_{max}
908
Realizar búsqueda
910
Proporcionar índice de vector de código de mejor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

Figura 9c

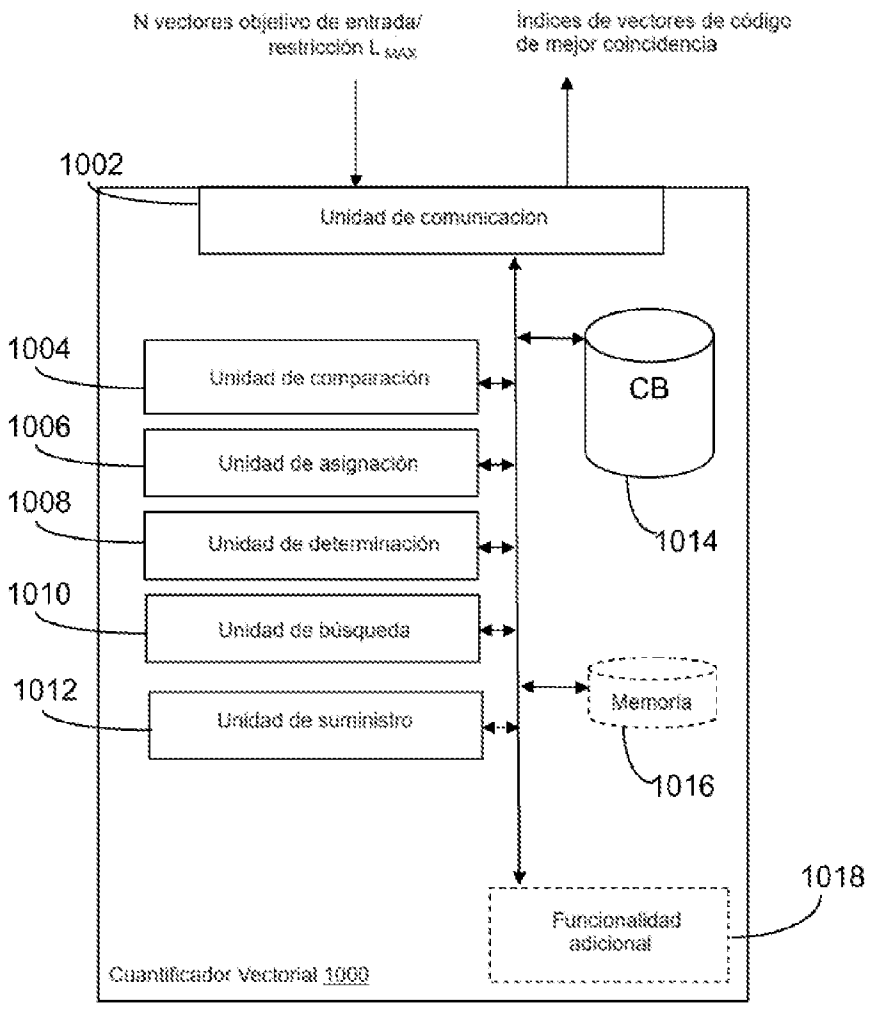
N vectores objetivo de entrada/ restricción L max
Índices de vectores de código de mejor coincidencia
1002
Unidad de comunicación
1004
Unidad de comparación
1006
Unidad de asignación
1008
Unidad de determinación
1010
Unidad de búsqueda
1012
Unidad de suministro
CB
1014
Memoria
1016
1018
Funcionalidad adicional
Cuantificador Vectorial 1000

Figura 10

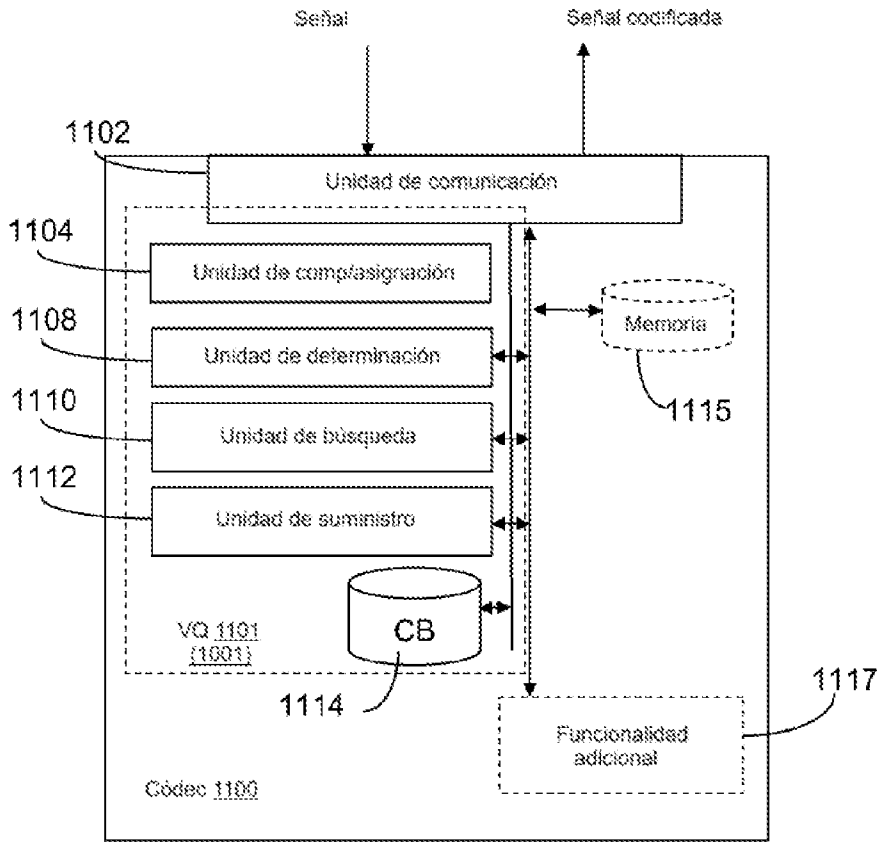
Señal
Señal codificada
1102
Unidad de comunicación
1104
Unidad de compdesignación
1108
Unidad de determinación
1110
Unidad de búsqueda
1112
Unidad de suministro
Memoria
1115
CB
VQ 1101
(1001)
1114
1117
Funcionalidad adicional
Códec 1100

Figura 11

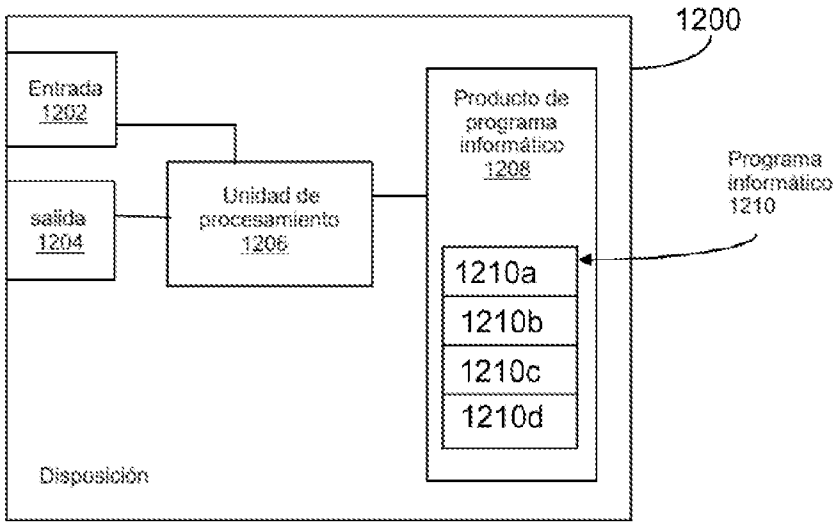
1200
Entrada
1202
salida
1204
Unidad de procesamiento
1206
Producto de programa informático
1208
Programa informático
1210
1210a
1210b
1210c
1210d
Disposición

Figura 12