(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 586 712**

(51) Int. Cl.:

G10L 19/038 (2013.01)
H03M 7/30 (2006.01)

(12) TRADUCCIÓN DE PATENTE EUROPEA T3

(96) Fecha de presentación y número de la solicitud europea: **25.06.2015** **E 15733942 (5)**

(97) Fecha y número de publicación de la concesión europea: **25.05.2016** **EP 2992529**

(54) Título: **Búsqueda de forma de cuantificador de vector en pirámide**

(30) Prioridad:

28.07.2014 US 201462029586 P

(45) Fecha de publicación y mención en BOPI de la traducción de la patente:
18.10.2016

(73) Titular/es:

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

(72) Inventor/es:

SVEDBERG, JONAS

(74) Agente/Representante:

DE ELZABURU MÁRQUEZ, Alberto

ES 2 586 712 T3

DESCRIPCIÓN

Búsqueda de forma de cuantificador de vector en pirámide

Campo de la invención

La descripción en la presente memoria se refiere a cuantificación de vector (VQ) realizada por un codificador.

Antecedentes

Es conocido que la cuantificación de vector sin restricciones es el método de cuantificación óptimo para muestras agrupadas, es decir, vectores, de una cierta longitud. No obstante, la implementación de cuantificación de vector sin restricciones implica altos requisitos en términos de complejidad y capacidad de memoria. Un deseo de permitir implementación de cuantificación de vector también en situaciones con restricciones de memoria y complejidad de búsqueda, ha conducido al desarrollo de los llamados cuantificadores de vector estructurados. Diferentes estructuras dan diferentes compromisos en términos de requisitos de complejidad de búsqueda y memoria. Un método tal es la llamada cuantificación de vector de ganancia-forma, donde el vector objetivo t se representa usando un vector de forma x y un valor de ganancia G:

$$x = \frac{t}{G} \qquad \text{(Ec. 0)}$$

El concepto de cuantificación de vector de ganancia-forma es para cuantificar el par $\{x, G\}$ en lugar de cuantificar directamente el vector objetivo t. Los componentes de la ganancia (G) y la forma (x) se codifican usando un cuantificador de forma que se sintoniza para la entrada de forma normalizada y un cuantificador de ganancia que maneja la dinámica de la señal. Esta estructura de ganancia-forma se usa frecuentemente en codificación de audio dado que la división en dinámica y forma, también denotada estructura fina, se ajusta bien con el modelo de auditoría perceptiva. El concepto de ganancia-forma también se puede aplicar a coeficientes de Transformada Coseno Discreta u otros coeficientes usados en codificación de vídeo.

Muchos códec de habla y audio tales como G.718 de la ITU-T y Opus (RFC 6716) del IETF usan una VQ de ganancia-forma basada en una PVQ estructurada a fin de codificar los coeficientes espectrales de la señal de habla/audio objetivo.

El concepto de codificación PVQ se introdujo por R. Fischer en el periodo de tiempo 1983-1986 y ha evolucionado a un uso práctico desde entonces con la llegada de Procesadores de Señal Digital, DSP, más eficientes. El concepto de codificación PVQ implica la búsqueda de localización y luego codificación de un punto en una hiperpirámide N-dimensional con la norma L1 de entero de K pulsos unidad. La denominada norma L1 es la suma de los valores absolutos del vector, es decir, la suma absoluta del vector de PVQ entero con signo se restringe a ser exactamente K, donde un pulso unidad se representa por un valor entero de “1”. Un entero con signo es capaz de representar enteros negativos, en relación a sin signo que solamente pueden representar enteros no negativos.

Uno de los beneficios de interés con el planteamiento de codificación PVQ estructurada a diferencia de muchas otras VQ estructuradas es que no hay límite inherente con respecto a la dimensión N, de manera que los métodos desarrollados para codificación PVQ deberían ser aplicables a cualquier dimensión N y a cualquier valor K.

Un problema con la cuantificación de forma PVQ estructurada es encontrar el mejor vector cuantificado posible usando una cantidad razonable de complejidad. Para codificación de habla y audio de tasa más alta, cuando el número de pulsos unidad permitido K, pueda llegar a ser muy alto y la dimensión N también pueda ser alta, hay demandas incluso más fuertes de tener una búsqueda de PVQ eficiente, al tiempo que se mantiene la calidad, por ejemplo, en términos de Relación Señal a Ruido, SNR, del habla/audio reconstruido.

Además, el uso del concepto PVQ no está restringido al área de codificación de habla y audio. Actualmente, el denominado Grupo de Trabajo de Ingeniería de Internet, IETF, está persiguiendo el desarrollo de un códec de vídeo donde los coeficientes de Transformada Coseno Discreta, DCT, se codifican usando un algoritmo basado en PVQ. En codificación de vídeo es incluso más importante que en codificación de audio tener un procedimiento de búsqueda eficiente, ya que el número de coeficientes puede llegar a ser muy grande con visualizadores grandes.

Una manera de realizar una cuantificación de vector en pirámide de la forma se describe en la sección 3.2 de Valin et al. “A full-bandwidth audio codec with low complexity and very low delay”, EUSIPCO, 2009.

Compendio

Para una PVQ estructurada se desea permitir una búsqueda computacionalmente eficiente que aún proporcione una relación Señal a Ruido alta. Especialmente para implementaciones que implican un DSP de precisión fija. La solución proporcionada en la presente memoria permite una búsqueda de forma de PVQ computacionalmente eficiente, proporcionando una búsqueda fina de PVQ mejorada.

Según un primer aspecto, se proporciona un método para búsqueda de forma de PVQ, a ser realizada por un codificador. La PVQ se supone que implica tomar el vector objetivo x como entrada que deriva un vector y añadiendo iterativamente pulsos unidad en un bucle de búsqueda de dimensión interior. El método proporcionado comprende, antes de entrar en un bucle de búsqueda de dimensión interior siguiente para adición de pulso unidad: determinar, en base a una amplitud de pulso máxima, $maxamp_y$, de un vector y actual, si se necesita más de una longitud de palabra de bit actual para representar, de una manera sin pérdidas, una variable, $enloop_y$. La variable $enloop_y$ que se relaciona con una energía acumulada de y, en el bucle de dimensión interior próximo.

Según un segundo aspecto, se proporciona un codificador, para búsqueda de forma de PVQ. La PVQ se supone que implica tomar un vector objetivo x como entrada que deriva un vector y añadiendo iterativamente pulsos unidad en un bucle de búsqueda de dimensión interior. El codificador proporcionado se configura para, antes de entrar en un siguiente bucle de búsqueda de dimensión interior para adición de pulso unidad: determinar, en base a una amplitud de pulso máxima, $maxamp_y$, de un vector y actual, si se necesita más de una longitud de palabra de bit actual para representar, de una manera sin pérdidas, una variable, $enloop_y$. La variable $enloop_y$ que se relaciona con una energía acumulada de y, en el bucle de dimensión interior próximo.

El método puede comprender y el codificador se puede configurar para, antes de entrar en el siguiente bucle de dimensión interior para adición de pulso unidad: determinar, en base a un valor absoluto máximo, $xabs_{max}$, del vector de entrada x, un cambio ascendente, en una palabra de bit, del siguiente valor de correlación dentro del bucle acumulado del siguiente bucle, $corr_{xy}$, entre x y el vector y.

El método puede comprender y el codificador se puede configurar para, cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$, realizar los cálculos de bucle interior usando una longitud de palabra de bit más larga para representar $enloop_y$.

El método puede comprender y el codificador se puede configurar para, cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$, realizar los cálculos de bucle interior usando una longitud de palabra de bit más larga para representar un valor de correlación dentro del bucle acumulada cuadrada, $corr_{xy}^2$, entre x y el vector y, en el bucle interior.

El método puede comprender además y el codificador se puede configurar para, cuando no se necesita más de una longitud de palabra de bit actual para representar $enloop_y$:

- realizar los cálculos de bucle interior empleando un primer bucle de adición de pulso unidad usando una primera longitud de palabra de bit para representar $enloop_y$ y:

cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$:

- realizar los cálculos de bucle interior empleando un segundo bucle de adición de pulso unidad usando una longitud de palabra de bit más larga para representar $enloop_y$ que el primer bucle de adición de pulso unidad.

La determinación, basada en $maxamp_y$, de si se necesita más de una longitud de palabra de bit actual para representar $enloop_y$ puede comprender determinar las características del caso cuando, en el bucle de búsqueda interior próximo, el pulso unidad se añade a la posición en y que está asociada con $maxamp_y$.

El método puede comprender además y el codificador se puede configurar para, en el bucle de búsqueda de dimensión interior para adición de pulso unidad:

- determinar una posición, n_{best}, en y para adición de un pulso unidad evaluando una multiplicación cruzada, para cada posición n en y, de un valor de correlación y de energía para la n actual; y un valor de correlación cuadrada, BestCorrSq y uno de energía, bestEn, guardados a partir de valores previos de n; como:

$$corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

donde

$$\left.\begin{aligned} n_{best} &= n \\ bestEn &= enloop_y \\ BestCorrSq &= corr_{xy}^2 \end{aligned}\right\} \text{cuando } corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

El método también puede comprender y el codificador se puede configurar para hacer el seguimiento de $maxamp_y$ cuando un valor final de K, asociado con el vector objetivo x, excede un valor umbral. Aquí el método puede comprender y el codificador se puede configurar para calcular un margen de energía, en_margin, solamente si un valor actual de K excede un valor umbral que puede ser el valor umbral mencionado en la frase precedente.

Según un tercer aspecto, se proporciona un dispositivo de comunicación, que comprende un codificador según el segundo aspecto.

Según un cuarto aspecto, se proporciona un programa de ordenador, que comprende instrucciones que, cuando se ejecutan en al menos un procesador, tal como un DSP, hacen al por lo menos un procesador llevar a cabo el método según el primer aspecto.

Según un quinto aspecto, se proporciona un portador, que contiene el programa de ordenador según el cuarto aspecto, el portador que es uno de una señal electrónica, señal óptica, señal radio o medio de almacenamiento legible por ordenador.

Breve descripción de los dibujos

Los precedentes y otros objetos, rasgos y ventajas de la tecnología descrita en la presente memoria serán evidentes a partir de la descripción más particular de las realizaciones que se ilustran en los dibujos anexos. Los dibujos no están necesariamente a escala, en su lugar, el énfasis se pone en ilustrar los principios de la tecnología descrita en la presente memoria.

Las figuras 1-4 ilustran un método para búsqueda de forma de PVQ (búsqueda fina), según diferentes realizaciones ejemplares.

La figura 5 muestra pasos de una realización de una búsqueda de forma de PVQ (búsqueda fina), según una realización ejemplar.

La figura 6 muestra pasos de la búsqueda de forma de PVQ (búsqueda fina) de la figura 5 en más detalle, según una realización ejemplar.

La figura 7 ilustra realizaciones de una búsqueda de forma de PVQ.

La figura 8 muestra una realización de un dispositivo de comunicación equipado con un codificador EVS.

La figura 9 muestra una realización de un dispositivo de comunicación y

La figura 10 también muestra una realización de un dispositivo de comunicación.

Las figuras 11a-c muestran un codificador según realizaciones ejemplares.

La figura 12 muestra un ejemplo de un sistema de codificación de audio de PVQ, donde al menos una parte del sistema está comprendida en un codificador y/o códec que a su vez está comprendido en un dispositivo de comunicación, tal como un teléfono móvil.

Descripción detallada

En aritmética de punto flotante no hay ningún problema importante relacionado con establecer la dinámica de parámetros de iteración de búsqueda de forma de PVQ de bucle interior, no obstante en los DSP de precisión fija por ejemplo con acumuladores (un registro en el que se almacenan resultados aritméticos y/o lógicos intermedios) y variables limitados de 16/32 bits, es muy importante emplear métodos de búsqueda eficientes donde se maximiza el rango dinámico limitado de las variables de DSP y se maximiza la precisión, mientras que es capaz de usar tantas de las operaciones de DSP de rango dinámico limitado rápidas disponibles como sea posible.

El término “precisión” anterior se refiere a ser capaz de representar tantos números como sea posible, es decir, el número de bits después del punto decimal para una longitud de palabra específica. Otra forma de decirlo es que la precisión corresponde a la resolución de la representación, que se define de nuevo por el número de decimales o dígitos binarios. La razón para que la precisión en las realizaciones descritas más adelante se pueda decir que correlaciona con el número de bits después del punto decimal y no necesariamente con la longitud de palabra en sí misma es que en aritmética de punto fijo, puede haber diferentes precisiones para la misma longitud de palabra. Por ejemplo, los formatos de datos 1 Q15 y 2 Q14 ambos tienen longitud de palabra 16, pero el primero tiene 15 bits después del punto decimal y el otro 14 bits. El menor número representable sería entonces 2^-15 y 2^-14 respectivamente.

Una manera de realizar cuantificación de vector en pirámide de la forma se describe en la sección 3.2 de Valin et al., “A full-bandwidth audio codec with low complexity and very low delay”, EUSIPCO, 2009. En este documento se presenta un códec MDCT donde los detalles, es decir, la forma, en cada banda se cuantifican algebraicamente usando un libro de códigos esférico y donde la asignación de bits se infiere a partir de información compartida entre el codificador y el decodificador. Aspectos y realizaciones de la descripción de esta solicitud al menos se refieren aproximadamente a cómo hacer una búsqueda según las Ecuaciones 4-7 en Valin et al., de una forma eficiente en punto fijo limitado a, por ejemplo, aritmética de 16/32 bits en lugar de valores flotantes como en Valin et. al.

En algunos aspectos y realizaciones descritas en lo sucesivo, dado un vector objetivo *x(n)* (*t* en la Ecuación 0) de cierta dimensión *N* y dado un cierto número de pulsos unidad *K*, se analiza la forma y se determina un vector de reconstrucción adecuado *xq(n)=func(y)n))*, que minimiza el error de cuantificación de forma y de esta manera maximiza una calidad percibida por ejemplo en caso de codificación de audio. Al menos algunos de los aspectos y realizaciones se implementan para aspirar al hallazgo de la constelación óptima de *K* pulsos unidad, en un vector *y(n)* que necesita adherirse a la norma L1, mientras que mantiene la complejidad bajo control, es decir, tan baja como sea posible en la práctica.

En lugar de usar los métodos de bucle abierto de la técnica anterior para determinar valores aproximados para el rango dinámico de bucle interior y precisión de acumulador, algunos de los aspectos y realizaciones se diseñan para usar un coste bajo, en términos de ciclos del DSP necesarios y en términos de Memoria Sólo de Lectura (ROM) de Programa adicional necesaria, análisis previo “casi óptimo” del numerador de peor caso y/o del denominador de peor caso antes de comenzar las evaluaciones costosas del cociente de distorsión de forma de PVQ en el bucle de búsqueda más interior. El análisis previo “casi óptimo” no se dirige a escalar los valores al rango dinámico máximo óptimo exacto, sino que en su lugar el análisis previo determina la potencia casi óptima de factor de 2 escalas, ya que la potencia de 2 escalas se puede implementar como cambios de un número binario y tales cambios tienen un bajo coste en ciclos del DSP y en ROM del DSP.

La selección de precisión del denominador se motiva perceptualmente como regiones con picos espectralmente que se asignarán con más precisión que regiones más planas.

Aunque algunos de los conceptos principales descritos en la descripción cubren diversas modificaciones y construcciones alternativas, las realizaciones de los aspectos se muestran en los dibujos y código ejemplar y se describirán en detalle en lo sucesivo.

Introducción de optimización general de búsqueda de PVQ

Un cuantificador de PVQ estructurado de norma L1 permite varias optimizaciones de búsqueda, donde una optimización primaria es mover el objetivo al “cuadrante” todo positivo (también se podría denotar ortante o hiperoctante) en un espacio N dimensional y una segunda optimización es usar una proyección de norma L1 como una aproximación de inicio para *y(n)*. Una norma L1 de *K* para una *PVQ(N,K)* significa que la suma absoluta de todos los elementos en el vector de PVQ *y(n)* tiene que ser K, justo como la suma absoluta de todos los elementos en el vector de forma objetivo x(n).

Una tercera optimización es actualizar iterativamente los términos del cociente Q_{PVQ} de $corr_{xy}^2$ y $energy_y$, en lugar de volver a calcular la Ec. 4 (de más adelante) sobre el espacio de vector entero N para cada cambio candidato al vector *y(n)* en la búsqueda de alcanzar el *K* de norma L1, que se requiere para el paso de indexación posterior.

Los tres pasos de optimización principales anteriores son optimizaciones que pueden existir generalmente en implementaciones de PVQ pasadas tales como CELT e IETF-Opus y parcialmente en la G.718, no obstante para la exhaustividad de la descripción de los aspectos y las realizaciones, estos pasos se perfilan también brevemente más adelante.

Búsqueda de forma del vector PVQ eficiente

Una descripción general de un sistema de codificación y decodificación de audio que aplica una realización de la búsqueda de forma de PVQ propuesta en la presente memoria se puede ver en la Fig. 12. Una búsqueda de forma general que usa una proyección en pirámide seguida por un flujo de búsqueda (de forma) fina se puede ver por ejemplo en la figura 5. Otra realización de una parte de búsqueda fina de una búsqueda de forma se representa en la figura 6. Una búsqueda de forma de PVQ puede comprender una proyección en pirámide y una búsqueda fina. Cuando no se aplica una proyección en pirámide, la búsqueda de forma solamente comprende la búsqueda fina. Por lo tanto, “búsqueda fina” y “búsqueda de forma” se pueden usar algunas veces intercambiablemente en la presente memoria, dado que la búsqueda fina es una parte de la búsqueda de forma y cuando no hay búsqueda tosca inicial, mediante proyección en pirámide, la realización de la búsqueda de forma es incluso la misma cosa que la realización de la búsqueda fina. En otras palabras, la búsqueda fina algunas veces puede ser o constituir la búsqueda de forma y cuando se aplica proyección en pirámide, la búsqueda fina es parte de la búsqueda de forma.

Introducción a la búsqueda de PVQ

La meta del procedimiento de búsqueda de *PVQ(N,K)* es encontrar el vector de salida escalado y normalizado $x_p(n)$ mejor. $x_p(n)$ se define como:

$$\mathbf{x_q} = \frac{\mathbf{y}}{\sqrt{\mathbf{y}^{\mathbf{T}}\mathbf{y}}} \qquad \text{(Ec. 1)}$$

Donde $y = y_{N,K}$ es un punto en la superficie de una hiperpirámide N dimensional y la norma L1 de $y_{N,K}$ es K. En otras palabras, $y_{N,K}$ es el vector de código de forma de entero seleccionado de tamaño N, también denotado dimensión N, según:

$$\mathbf{y}_{N,K} = \left\{ e : \sum_{i=0}^{N-1} \left| e_i \right| = K \right\} \qquad \text{(Ec. 2)}$$

Es decir, el vector x_q es el subvector entero normalizado de energía unidad $y_{N,K}$. El vector y mejor es el que minimiza el error de forma cuadrático medio entre el vector objetivo $x(n)$ y el vector de salida cuantificado normalizado escalado x_q. Esto se logra minimizando la siguiente distorsión de búsqueda:

$$d_{PVQ} = -\mathbf{x}^T \mathbf{x_q} = -\frac{\left(\mathbf{x}^T \mathbf{y}\right)}{\sqrt{\mathbf{y}^\mathbf{T} \mathbf{y}}} \qquad \text{(Ec. 3)}$$

O de manera equivalente, elevando al cuadrado el numerador y denominador, maximizando el cociente Q_{PVQ}:

$$Q_{PVQ} = \frac{\left(\mathbf{x}^T \mathbf{y}\right)^2}{\mathbf{y}^\mathbf{T} \mathbf{y}} = \frac{\left(corr_{xy}\right)^2}{energía_{\,y}} \qquad \text{(Ec. 4)}$$

donde $corr_{xy}$ es la correlación entre x e y. En la búsqueda de la forma de vector de PVQ óptima $y(n)$ con norma L1 de K, las actualizaciones iterativas de las variables de Q_{PVQ} se hacen en el "cuadrante" todo positivo en un espacio N dimensional según:

$$corr_{xy}(k,n) = corr_{xy}(k-1) + 1 \cdot x(n) \qquad \text{(Ec. 5)}$$

$$energy_y(k,n) = energy_y(k-1) + 2 \cdot 1^2 \cdot y(k-1,n) + 1^2 \qquad \text{(Ec. 6)}$$

donde $corr_{xy}(k\text{-}1)$ significa la correlación lograda hasta ahora colocando $k\text{-}1$ pulsos unidad previos y $energy_y(k\text{-}1)$ significa la energía acumulada lograda hasta ahora colocando $k\text{-}1$ pulsos unidad e $y(k\text{-}1, n)$ significa la amplitud de y en la posición n a partir de la colocación previa de $k\text{-}1$ pulsos unidad. Para acelerar aún más el procesamiento iterativo dentro del bucle el término de energía $energy_y(k)$ se reduce a 2, eliminando de esta manera una multiplicación en el bucle interior.

$$\begin{aligned} enloop_y(k,n) &= energy_y(k,n)/2, \Rightarrow \\ enloop_y(k,n) &= enloop_y(k-1) + y(k-1,n) + 0.5 \end{aligned} \qquad \text{(Ec. 7)}$$

donde $enloopy(k,n)$ es la variable de energía preferida usada y acumulada dentro del bucle de búsqueda de pulso unidad más interior, ya que su actualización iterativa requiere una multiplicación menor que $energy_y(k,n)$.

$$Q_{PVQ}(k,n) = \frac{corr_{xy}(k,n)^2}{enloop_y(k,n)} \qquad \text{(Ec. 8)}$$

La posición mejor n_{best} para el pulso unidad de orden k, se actualiza iterativamente aumentando n desde 0 a $N\text{-}1$:

$$n_{best} = n \quad , si \quad Q_{PVQ}(k,n) > Q_{PVQ}(k,n_{best})$$

(Ec. 9)

Para evitar divisiones costosas, lo cual es especialmente importante en aritmética de punto fijo, la decisión de actualización de maximización de Q_{PVQ} se realiza usando una multiplicación cruzada del numerador de correlación cuadrada mejor *bestCorrSq* guardado y el denominador de energía mejor *bestEn* guardado hasta ahora, que se podría expresar como:

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq = corr_{xy}(k,n)^2 \\ bestEn = enloop_y(k,n) \end{array}\right\}, si \quad corr_{xy}(k,n)^2 \cdot bestEn > bestCorrSq \cdot enloop_y(k,n)$$

(Ec. 10)

La maximización iterativa de $Q_{PVQ}(k,n)$ puede comenzar a partir de un número cero de pulsos unidad colocados o a partir de un número de colocación previa de coste menor adaptativo de pulsos unidad, en base a una proyección entera a un punto por debajo de la superficie de la pirámide de orden *K*, con un subimpulso garantizado de pulsos unidad en la norma L1 objetivo *K*.

Análisis de preparación de búsqueda de PVQ

Debido a la naturaleza estructurada del vector entero de PVQ $y_{N,K}$, donde se permiten todas las combinaciones de signos posibles y es posible codificar todas las combinaciones de signos, siempre que el vector resultante se adhiera a la norma L1 de K pulsos unidad, la búsqueda se realiza en el primer "cuadrante" todo positivo (la razón para las comillas en "cuadrante" es que un cuadrante verdadero solamente existe cuando N=2 y N puede ser aquí más de 2). Además, como se ha dado cuenta el inventor, para lograr tan alta precisión como sea posible para una implementación de precisión limitada, el valor absoluto máximo $xabs_{max}$ de la señal de entrada *x(n)* se puede analizar previamente para uso futuro en la configuración del procedimiento de acumulación de correlación de bucle interior.

$$xabs(n) = |\mathbf{x}(n)|, para \quad n = 0,\ldots,N-1 \qquad \text{(Ec. 11)}$$

$$xabs_{max} = max(xabs_0, \ldots, xabs_{N-1}) \qquad \text{(Ec. 12)}$$

Manejo de objetivos de energía muy baja y subvectores de energía muy baja

En caso de que el vector objetivo de entrada (*x* en la Ec. 3 o *t* en la Ec. 0) sea un vector todo ceros y/o la ganancia del vector (por ejemplo, *G* en la Ec. 0) sea muy baja, la búsqueda de PVQ se puede dejar de lado y un vector *y* de PVQ válido se puede crear de manera determinística asignando la mitad de los *K* pulsos unidad a la primera posición $\left(y[0] = \left\lfloor \frac{K}{2} \right\rfloor\right)$ y los pulsos unidad restantes a la última posición $(y[N-1]=y[N-1]+(K-y[0]))$.

El término "objetivos de energía muy baja" y "ganancia de vector muy baja" es una realización tan baja como cero, que se ilustra en el código ANSI C ejemplar descrito más adelante, donde el código correspondiente es:

```
IF( L_xsum == 0  ||  neg_gain == 0  )

{   /* caso de entrada cero o ganancia cero */
```

No obstante, también puede ser menor o igual que épsilon o EPS, donde EPS es el valor más bajo que es mayor que cero y que se considera como que es digno de que se represente en una precisión seleccionada. Por ejemplo, en una precisión Q15 en una palabra de 16 bits con signo, la ganancia de subvector llega a ser menor o igual que EPS 1/2^15= 1/32.768 (por ejemplo, una ganancia de vector menor o igual que 0,000030517578125) y en caso de precisión Q12 en una palabra de 16 bits con signo para el vector objetivo x(n), entonces el valor "muy bajo" llega a ser EPS = (1/2^12), por ejemplo sum(abs(x(n))) menor o igual que 0,000244140625. En una realización de aritmética de DSP de punto fijo con palabra de 16 bits, un formato de entero sin signo puede tomar cualquier valor entero desde 0 a 65.546, en donde un entero con signo puede tomar el valor de -32.768 a +32.767. Usando un formato de fracción sin signo, los 565.536 niveles se extienden uniformemente entre 0 y +1, mientras que en una realización de formato de fracción con signo los niveles se separarían por igual entre -1 y +1.

Aplicando este paso opcional relacionado con vectores cero y valores de ganancia baja, se reduce la complejidad de búsqueda de PVQ y la complejidad de indexación se extiende/comparte entre el codificador que indexa y decodificador que desindexa, es decir, no se "gasta" ningún procesamiento para buscar un vector objetivo cero o un vector objetivo muy bajo que se podría reducir de cualquier forma a cero.

Proyección de búsqueda previa de PVQ opcional

Si la relación de densidad de pulso *K/N* es mayor que 0,5 pulsos unidad por coeficiente, por ejemplo coeficiente de transformada coseno discreta modificada, se hace una proyección de bajo coste a la subpirámide *K-1* y se usa como un punto de inicio para *y*. Por otra parte, si la relación de densidad de pulso es menor que 0,5 pulsos unidad por coeficiente, la búsqueda de PVQ iterativa comenzará desde *0* pulsos unidad colocados previamente. La proyección de bajo coste a "K-1" es típicamente menos cara computacionalmente en ciclos del DSP que repetir la búsqueda de bucle interior de pulso unidad *K-1* veces. No obstante, un inconveniente de la proyección de bajo coste es que producirá un resultado inexacto debido a la aplicación de la función suelo N dimensional. La norma L1 resultante de la proyección de bajo coste que usa la función suelo típicamente puede ser cualquiera entre "*K-1*" a aproximadamente "*K-5*", es decir, el resultado después de la proyección necesita ser buscado de manera fina para alcanzar la norma objetivo de *K*.

La proyección de bajo coste se realiza como:

$$proj_{fac} = \frac{K-1}{\sum_{n=0}^{n=N-1} \mathbf{xabs}(n)} \qquad \text{(Ec. 13)}$$

$$y(n) = y_{start}(n) = \lfloor \mathbf{xabs}(n) \cdot proj_{fac} \rfloor \text{ para n=0,..,N-1} \qquad \text{(Ec. 14)}$$

Si no se hace ninguna proyección, el punto de inicio es un vector *y(n)* todo de ceros. El coste del DSP de la proyección en ciclos del DSP está en los alrededores de *N*(suma absoluta)+25(la división)+*2N*(multiplicación y suelo) ciclos.

En la preparación de la búsqueda fina para alcanzar la superficie de la pirámide de orden *K* el número acumulado de pulsos unidad $pulse_{tot}$, la correlación acumulada $corr_{xy}(pulse_{tot})$ y la energía acumulada $energy_y(pulse_{tot})$ para el punto de inicio se calculan como:

$$pulse_{tot} = \sum_{n=0}^{n=N-1} \mathbf{y}(n) \qquad \text{(Ec. 15)}$$

$$corr_{xy}(pulse_{tot}) = \sum_{n=0}^{n=N-1} \mathbf{y}(n) \cdot \mathbf{xabs}(n) \qquad \text{(Ec. 16)}$$

$$energy_y(pulse_{tot}) = \sum_{n=0}^{n=N-1} \mathbf{y}(n) \cdot \mathbf{y}(n) = \|\mathbf{y}\|_{L2} \qquad \text{(Ec. 17)}$$

$$enloop_y(pulse_{tot}) = energy_y(pulse_{tot})/2 \qquad \text{(Ec. 18)}$$

Búsqueda fina de PVQ

La solución descrita en la presente memoria se relaciona con la búsqueda fina de PVQ (que constituye o es parte de la búsqueda de forma de PVQ, como se describió previamente). Lo que se ha descrito en las secciones precedentes es principalmente la PVQ de la técnica anterior, excepto para la determinación por adelantado de $xabs_{max}$, que se describirá además más adelante. El vector de forma entero final *y(n)* de dimensión N debe adherirse a la norma L1 de *K* pulsos. La búsqueda fina se supone que está configurada para comenzar desde un punto más bajo en la pirámide, es decir por debajo de la pirámide de orden K y encontrar iterativamente su forma para la superficie de la hiperpirámide de orden *K N* dimensional. El valor *K* en la búsqueda fina puede oscilar típicamente de 1 a 512 pulsos unidad.

El inventor se ha dado cuenta, que a fin de mantener la complejidad de la búsqueda e indexación de PVQ a un nivel razonable, la búsqueda se puede dividir en dos ramales principales, donde un ramal se usa cuando se conoce que la representación de energía dentro del bucle de *y(n)* permanecerá dentro de una palabra de 16 bits con signo o sin signo durante una siguiente iteración de bucle de búsqueda interior y otro ramal se usa cuando la energía dentro del bucle puede exceder el rango dinámico de una palabra de 16 bits durante una siguiente iteración de bucle de búsqueda interior.

Búsqueda fina de precisión fija para un número bajo de pulsos unidad

Cuando el *K* final es menor o igual que un umbral de t_p=127 pulsos unidad, la dinámica de la $energy_y(K)$ siempre permanecerá dentro de 14 bits y la dinámica del cambio ascendente de 1 bit $enloop_y(K)$ siempre permanecerá dentro de 15 bits. Esto permite el uso de una palabra de 16 bits con signo para representar cada $enloop_y(k)$ dentro de todas las iteraciones de bucle interior de búsqueda de pulso fina hasta *k=K*. En otras palabras, no habrá necesidad de una longitud de bit de palabra que exceda 16 bits para representar $energy_y(K)$ o $enloop_y(K)$ en cualquier iteración de bucle interior de búsqueda de pulso fina cuando K<127.

En el caso de la disponibilidad de los operadores eficientes Multiply, MultiplyAdd(multiplicar-sumar) y MutiplySubstract (multiplicar-restar) de DSP para variables de 16 bits sin signo, el umbral se puede aumentar a t_p = 255, ya que entonces $enloop_y(K)$ permanecerá siempre dentro de una palabra de 16 bits sin signo. MultiplyAdd es aquí en una realización instrucciones multiplicar-sumar u operaciones equivalentes para multiplicar valores de datos que representan señales de audio y vídeo por valores de filtro o transformación y acumular los productos para reproducir un resultado. Las operaciones MultiplySubstract son las mismas que las operaciones MultiplyAdd, excepto que las sumas se sustituyen por restas.

En la preparación de la siguiente adición de pulso unidad, el cambio ascendente máximo posible casi óptimo del valor de correlación dentro del bucle acumulado del siguiente bucle, $corr_{xy}$, en una palabra de 32 bits con signo se analiza previamente usando el valor de entrada máximo absoluto calculado previamente $xabs_{max}$ como:

$$corr_{upshift} = 31 - \left\lfloor \log2\left(corr_{xy}(pulse_{tot}) + 2 \cdot (1 \cdot xabs_{max})\right)\right\rfloor \quad \text{(Ec. 19)}$$

El cambio ascendente calculado en la Ec. 19 representa el “peor caso” y cubre el cambio ascendente máximo posible que se puede hacer en el siguiente bucle interior y de esta manera asegura que la información más significativa relacionada con correlación no se perderá o desplazará hacia fuera, durante la interacción de bucle interior, incluso para el escenario del peor caso.

Este análisis dinámico de bucle interior previo de peor caso se puede realizar en 2-3 ciclos en la mayoría de las arquitecturas de DSP usando instrucciones MultiplyAdd y Norm (normalización) y el análisis es siempre el mismo independiente de la dimensión N. En un DSP de 16/32 bits virtual de la G.191 de la ITU-T las operaciones en la Ec. 19 llegan a ser: “corr_upshift = norm_l(L_nnac(*L_corrxy,1, xabs_max));” con un coste de 2 ciclos. Se debería señalar que norm_l(x) aquí corresponde a “31 – ceil(log2(x))” y se podría denotar alternativamente 31 – ceil(log2(x)), donde ceil(x) es la denominada función techo, que correlaciona un número real al entero siguiente más pequeño. Con más precisión, ceiling(x)=$\lceil x \rceil$ es el entero más pequeño no menor que *x*. Para $corr_{upshift}$ el término dentro de los corchetes con la barra horizontal superior es siempre un número positivo. La $corr_{upshift}$ se podría calcular alternativamente usando una función suelo como:

$$corr_{upshift} = 30 - \left\lfloor \log2\left(corr_{xy}(pulse_{tot}) + 2 \cdot (1 \cdot xabs_{max})\right)\right\rfloor$$

donde floor(x) = $\lfloor x \rfloor$ es el entero más grande no mayor que x.

Otro beneficio del planteamiento sugerido en la presente memoria para escalado de correlación de búsqueda de forma casi óptima es que el método propuesto no requiere un vector objetivo normalizado previamente x, lo cual ahorrará alguna complejidad adicional antes de comenzar la búsqueda de forma.

Para hacer la actualización de la Ec. 10 iterativa tan eficiente como sea posible, el *numerador* $corr_{xy}(k,n)^2$ se puede representar por una palabra con signo de 16 bits, incluso cuando se comprende más información que encaja en una palabra de 16 bits, mediante el siguiente planteamiento.

$$corr_{xy16}(k,n)^2 = \left(\mathrm{Round}_{16}\left(\left(corr_{xy}(pulse_{tot}) + 2 \cdot \left(1 \cdot \mathbf{xabs}(n)\right)\right) \cdot 2^{corr_{upshift}} \right)\right)^2$$

(Ec. 20)

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq_{16} = corr_{xy16}(k,n)^2 \\ bestEn_{16} = enloop_y(k,n) \end{array}\right\}, \text{ si } \quad corr_{xy16}(k,n)^2 \cdot bestEn_{16} > bestCorrSq_{16} \cdot enloop_y(k,n)$$

(Ec. 21)

donde la función "Round_{16}" extrae los 16 bits superiores de una variable de 32 bits con signo con redondeo. El cambio ascendente casi óptimo (Ec. 10) y el uso de una representación de 16 bits de la correlación cuadrada $bestCorrSq_{16}$ permite una búsqueda de bucle interior muy rápida usando solamente ~9 ciclos para realizar la prueba de la Ec. 21 y las tres actualizaciones variables, cuando se usan unas funciones Multiply, MultiplyAdd, MultiplySubstract optimizadas de los DSP.

La ubicación del siguiente pulso unidad en el vector y se determina ahora iterando sobre las *n=0,…, N-1* posiciones posibles en el vector y, mientras que se emplean las ecuaciones Ec. 20, Ec. 6 y Ec. 21.

Cuando se ha determinado la posición mejor n_{best} para el pulso unidad (en el vector y logrado hasta ahora), se actualizan la correlación acumulada $corr_{xy}(k)$, la energía dentro del bucle acumulada $enloop_y(k)$ y el número de pulsos unidad acumulados $pulse_{tot}$.

Si hay pulsos unidad adicionales para añadir, es decir, cuando $pulse_{tot} < K$, un nuevo bucle interior se inicia con un nuevo análisis $corr_{upshift}$ casi óptimo (Ec. 19) para la adición de un siguiente pulso unidad.

En total, este planteamiento sugerido tiene una complejidad de peor caso para cada pulso unidad añadido a *y(n)* de aproximadamente 5/N+15 ciclos por coeficiente cuantificado. En otras palabras, un bucle sobre un vector de tamaño N para añadir un pulso unidad tiene una complejidad de peor caso de alrededor de N*(5/N+15) ciclos, es decir, 5+15*N ciclos.

Búsqueda fina de precisión fija para un número alto de pulsos unidad

Cuando *K* es mayor que un umbral t_p, el cual en esta realización ejemplar que supone un DSP restringido de 16/32 bits, es t_p=127 pulsos unidad, la dinámica del parámetro $energy_y(K)$ puede exceder 14 bits y la dinámica del cambio ascendente de 1 bit $enloop_y(K)$ puede exceder 15 bits. De esta manera, a fin de no usar precisión innecesariamente alta, la búsqueda fina se configura para elegir adaptativamente entre representación de 16 bits y representación de 32 bits del par $\{corr_{xy}(k,n)^2, enloop_y(k,n)\}$ cuando *K* es mayor que t_p. Cuando *K* para el vector *y(n)* se conoce que termina en un valor final mayor que 127 por adelantado, la búsqueda fina hará el seguimiento de la amplitud de pulso máxima $maxamp_y$ en *y* lograda hasta ahora. Esto también se puede conocer a medida que se determina esa $maxamp_y$. Esta información de amplitud de pulso máxima se usa en un paso de análisis previo antes de entrar en el bucle de dimensión interior optimizado. El análisis previo comprende la determinación de qué precisión se debería usar para el bucle interior de adición de pulso unidad próximo. Como se muestra en la figura 12 mediante la entrada de N, K para la búsqueda de forma de PVQ, la asignación de bits se conoce/determina antes de que se inicie la búsqueda de PVQ. La asignación de bit puede usar fórmulas o almacenar tablas para obtener, determinar y/o calcular el K a ser introducido a la búsqueda de forma de PVQ, por ejemplo, K= function(bits(band),N) para una cierta banda con la dimensión N y un cierto número de bits(banda).

bits requeridos para PVQ(N,K)	N=8	N=16	N=32
K=4	11,4594	15,4263	19,4179
K=5	13,2021	18,1210	23,1001
K=6	14,7211	20,5637	26,5222
K=7	16,0631	22,7972	29,7253

Por ejemplo, una tabla almacenada como la mostrada anteriormente se puede usar para determinar o seleccionar un valor de K. Si la dimensión *N* es 8 y los bits disponibles para la banda *bits(band)* es *14,0*, entonces *K* se seleccionará para ser 5, ya que PVQ(N=8,K=6) requiere 14,7211 bits lo cual es mayor que el número de bits disponibles 14,0.

Si el análisis previo indica que se necesita más de una palabra de 16 bits con signo para representar la energía dentro del bucle sin perder ninguna información de energía, se emplea un bucle de adición de pulso unidad de precisión más alta y precisión alta computacionalmente más intensiva, donde tanto el termino de correlación cuadrada mejor guardado como el término de energía acumulada mejor guardado se representan por palabras de 32 bits.

$$en_{margin} = 31 - \left\lceil \log2\left(\left(1 + energy_y(pulse_{tot}) + 2 \cdot \left(1 \cdot maxamp_y\right)\right)\right)\right\rceil$$

(Ec. 22)

$$highprecision_{active} = \text{FALSA}, \qquad si\ (en_{margin} \geq 16)$$
$$highprecision_{active} = \text{VERDADERA}, \qquad si\ (en_{margin} < 16)$$

(Ec. 23)

El análisis dinámico de bucle interior previo de peor caso se puede realizar en 5-6 ciclos adicionales en la mayoría de los DSP y el coste del análisis es el mismo para todas las dimensiones. En un DSP de 16/32 bits virtual de STL 2009 de la G.191 de la ITU-T las operaciones en la Ec. 22 y Ec. 23 llegan a ser:

```
"L_energy_y    = L_add(L_energy_y, 1);  /* añadido 0,5 */
en_margin        = norm_l(L_mac(L_energy_y, 1, maxamp_y));
highprecision_active= 1;                move16();
if(sub(16,en_margin <= 0){
highprecision_active = 0;            move16();
}",
```

con un coste de máximo 6 ciclos.

El código correspondiente en un ejemplo de código ANSI-C más adelante es:

```
L_yy =L_add(L_yy,1);  /* añadido 0,5 */
en_margin =norm_l(L_mac(L_yy,1, max_amp_y)); /* "adición" máxima afinada, margen, ~2op.*/
en_dn_shift=sub(16, en_margin);  /* calc. cambio a palabra inferior */
high_prec_active = 1;  move16();
if( en_dn_shift <=0) { /* usar solamente energía de 32 bits si se necesita realmente */
high_prec_active = 0;  move16();
}
```

Alternativamente el margen de energía *en_margin* en la Ec.(22) se podría calcular en línea con una operación de la función STL de la G.191 *norm_l()* usando la función suelo como:

$$en_{margin} = 30 - \left\lfloor \log2\left(\left(1 + energy_y(pulse_{tot}) + 2 \cdot \left(1 \cdot maxamp_y\right)\right)\right)\right\rfloor$$

Si $highprecision_{active}$ es FALSA, es decir, =0, se emplea el bucle de búsqueda interior de menor precisión en la Ec. 20, Ec. 6 y Ec. 21, por otra parte, cuando $highprecision_{active}$ es VERDADERA, es decir, =1, la ubicación del siguiente pulso unidad se realiza empleando un bucle interior de mayor precisión, que representa $enloop_y$ y $corr_{xy}^2$ con palabras de 32 bits en este ejemplo. Es decir, cuando $highprecision_{active}$ es VERDADERA, la ubicación del siguiente pulso unidad en y(n) se determina iterando sobre *n=0,…, N-1* posiciones posibles, usando las ecuaciones Ec. 24, Ec. 6 y Ec. 25.

$$corr_{xy32}(k,n)^2 = \left(\left(corr_{xy}(pulse_{tot}) + 2 \cdot \left(1 \cdot \mathbf{xabs}(n)\right)\right) \cdot 2^{corr_{upshift}} \right)^2$$

(Ec. 24)

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq_{32} = corr_{xy32}(k,n)^2 \\ bestEn_{32} = enloop_y(k,n) \end{array}\right\}, si \quad corr_{xy32}(k,n)^2 \cdot bestEn_{32} > bestCorrSq_{32} \cdot enloop_y(k,n)$$

(Ec. 25)

En otras palabras, en_margin es indicativo de cuántos cambios ascendentes se pueden usar para normalizar la energía en el siguiente bucle. Si se pueden usar 16 o más cambios ascendentes, entonces la energía permanece en la longitud de palabra inferior, suponiendo longitudes de palabra de 16/32 bits y no hay necesidad del bucle de precisión alta (representación de 32 bits), así $highprecision_{active}$ se fija a FALSA. Una razón de implementación para hacerlo de esta manera (permitiendo que la información de energía permanezca en la parte baja de la palabra de 32 bits L_energy) es que es computacionalmente más barato: cuesta solamente 1 ciclo calcular extract_l(L_energy) mientras que un round_fx(L_shl(L_energy,en_margin)) alternativo lleva dos ciclos.

Cuando se ha determinado la posición mejor n_{best} del pulso unidad, la correlación acumulada $corr_{xy}(k)$, se actualizan la energía dentro del bucle acumulada $enloop_y(k)$ y el número de pulsos unidad acumulados $pulse_{tot}$. Además la amplitud máxima $maxamp_y$ en el mejor vector y entero hasta ahora, se mantiene actualizada, es decir, determinada, para el siguiente bucle de adición de pulso unidad.

$$maxamp_y = \max\left(maxamp_y, y\left[n_{best}\right]\right)$$

(Ec. 26)

Si además hay pulsos unidad a añadir, es decir, cuando $pulse_{tot} < K$, se inicia un nuevo bucle interior con un nuevo análisis $corr_{upshift}$ casi óptimo Ec. 19 y un nuevo análisis de precisión de energía Ec. 22 y Ec. 23 y entonces se comienza el siguiente bucle de pulso unidad con las ecuaciones Ec. 24, Ec. 6 y Ec. 26.

La complejidad del peor caso del planteamiento de alta precisión (en este ejemplo palabras de 32 bits) para cada pulso unidad añadido a *y(n)* es aproximadamente de 7/N+31 ciclos por coeficiente cuantificado.

El efecto de la selección de precisión de bucle interior basada en energía acumulada dentro del bucle es que los subvectores objetivo que tienen unos picos altos o tienen una granularidad muy fina, es decir, el *K* final es alto, estarán usando el bucle de precisión más alta y más ciclos más a menudo, mientras que los subvectores sin picos o de granularidad de pulsos baja usarán más a menudo el bucle de precisión menor y menos ciclos.

Se debería señalar que el análisis descrito en la sección anterior se podría realizar también cuando $K<t_p$. No obstante, se puede hacer una realización más eficiente mediante la introducción de un umbral t_p para aplicar el análisis anterior.

Finalización y normalización del vector de PVQ

Después de la búsqueda de forma, se asigna a cada elemento de vector de PVQ no cero su signo adecuado y el vector se normaliza L2 (también conocida como normalización Euclidiana) a energía unidad. Adicionalmente, si la banda se dividió, se escala además con una ganancia de subvector.

$$if\left(\mathbf{y}(n) > 0\right) \cap \left(\mathbf{x}(n) < 0\right) \Rightarrow \mathbf{y}(n) = -\mathbf{y}(n), para \quad n = 0,\ldots,N-1$$

(Ec. 27)

$$norm_{gain} = \frac{1}{\sqrt{\mathbf{y}^{\mathbf{T}}\mathbf{y}}}$$

(Ec. 28)

$$\mathbf{x_q}(n) = norm_{gain} \cdot \mathbf{y}(n), para \quad n = 0,\ldots,N-1$$

(Ec. 29)

Anteriormente, se presentaron y especificaron dos metodologías de precisión:

"En16 x CorrSq16", como se define en la sección anterior, (ecuaciones 19 hasta 21) y

"En32 x CorrSq32", (ecuaciones 22 hasta 26). Se describen más adelante dos métodos adicionales de complejidad media donde se varían la precisión del término de numerador Correlación Cuadrada y el término Energía.

Métodos "En16 x CorrSq32" y "En32 x CorrSq16"

El método *"En16 x CorrSq32"* es similar al *"En32 x CorrSq32"*, pero con la diferencia de que la actualización de pulso unidad mejor encontrada del bucle interior y la comparación usa una representación de 16 bits de la Energía mejor $bestEn_{16}$ hasta ahora, según:

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq_{32} = corr_{xy32}(k,n)^2 \\ bestEn_{16} = enloop_y(k,n) \end{array}\right\}, \text{ si } \; corr_{xy32}(k,n)^2 \cdot bestEn_{16} > bestCorrSq_{32} \cdot enloop_y(k,n)$$

(Ec. 30)

El coste aproximado del método *"En16 x CorrSq32"* por pulso unidad es 5/N + 21 ciclos.

El método *"En32 x CorrSq16"* es similar al *"En32 x CorrSq32"*, pero con la diferencia de que la actualización de pulso unidad mejor encontrada del bucle interior y la comparación usa una representación de 16 bits de la correlación cuadrada mejor $bestCorrSq_{16}$ hasta ahora, según:

$$\left.\begin{array}{c} n_{best} = n \\ bestCorrSq_{16} = corr_{xy16}(k,n)^2 \\ bestEn\,32 = enloop_y(k,n) \end{array}\right\}, \text{ si } \; corr_{xy16}(k,n)^2 \cdot bestEn\,32 > bestCorrSq\,16 \cdot enloop_y(k,n)$$

(Ec. 31)

El coste aproximado del método *"En32 x CorrSq16"* por adición de pulso unidad es 6/N + 20 ciclos por coeficiente.

Aspectos y realizaciones ejemplares

Más adelante, se describirán con referencia a las figuras 1-4 algunas realizaciones ejemplares de la solución descrita en la presente memoria.

La figura 1 es un diagrama de flujo que ilustra un método que concierne a una búsqueda fina de búsqueda de forma de PVQ. El método pretende que sea realizado por un codificador, tal como el codificador de medios para señales de audio y/o vídeo. El PVQ toma un vector de forma objetivo x como entrada y deriva un vector y añadiendo iterativamente pulsos unidad en un bucle de búsqueda de dimensión interior. El método se refiere a un análisis previo, que se hace antes de entrar en el bucle interior. Un vector de salida x_q entonces se derivará en base al vector y, como se describió previamente. No obstante, la formación de x_q no es central para la solución descrita en la presente memoria y por lo tanto no se describirá aún más aquí.

En el método ilustrado en la figura 1, se puede suponer que el codificador hace el seguimiento de un valor $maxamp_y$ de un vector y actual. Por "vector y actual" se entiende aquí el vector y compuesto, encontrado o construido hasta ahora, es decir para un k < K. Como se describió previamente, un punto de inicio para el vector y puede ser una proyección a una superficie por debajo de la pirámide de orden K o un vector de todo ceros vacío. El método ilustrado en la figura 1 comprende, antes de entrar en un próximo bucle de búsqueda de dimensión interior para adición de pulso unidad, determinar 101, en base a la amplitud de pulso máxima, $maxamp_y$, de un vector y actual, si se necesita más de una longitud de palabra de bit actual para representar $enloop_y$, de una manera sin pérdida en el bucle de dimensión interior próximo. La variable $enloop_y$ se refiere a una energía acumulada del vector y. La realización de este método permite al codificador mantener la complejidad de la búsqueda en un nivel razonable. Por ejemplo, permite al codificador aplicar un bucle de precisión aumentada (que implica una complejidad más alta) solamente cuando se pueda necesitar, analizando si el "escenario del peor caso" en el bucle interior próximo requeriría un bucle interior con una precisión más alta que la usada actualmente.

El análisis previo descrito anteriormente se realiza antes de cada entrada 102 al bucle interior, es decir, antes de cada adición de un pulso unidad al vector y. En una realización ejemplar donde solamente están disponibles dos representaciones de bit diferentes, es decir, longitudes de palabra de bit tales como 16 y 32 bits, el bucle interior se realizará usando una representación de 16 bits de $enloop_y$ hasta que se determine que se necesita una palabra de bit más larga para representar $enloop_y$, después de lo cual la longitud de palabra de bit más alta, es decir, la representación de 32 bits se aplicará para los cálculos de bucle interior. El bucle que usa una representación de 16 bits puede ser referido como un "bucle de precisión baja" y el bucle que usa una representación de 32 bits puede ser referido como un "bucle de precisión alta".

La determinación 102 de si se necesita más de una longitud de palabra de bit inicial o actual se podría expresar alternativamente como que se determina que la longitud de palabra de bit, de entre al menos dos longitudes de palabra de bit diferentes, alternativas, que se requerirán para representar el "peor caso" (aumento mayor posible) $enloop_y$ durante el siguiente bucle interior. Las al menos dos longitudes de palabra de bit diferentes podrían comprender al menos por ejemplo longitudes de palabra de 16 y 32 bits.

En otras palabras, cuando se determina 102 que se necesita más de una longitud de palabra de bit actual para representar $enloop_y$ en el siguiente bucle interior, los cálculos de bucle interior se realizan 103 con una longitud de palabra de bit más larga, que una longitud de palabra de bit inicial o actual, para representar $enloop_y$ en el bucle interior. Por otra parte, cuando se determina que no se necesita más de una longitud de palabra de bit para representar $enloop_y$, los cálculos de bucle interior se pueden realizar empleando un primer bucle de adición de pulso unidad que usa una longitud de palabra de bit primera o actual para representar $enloop_y$, es decir, la longitud de palabra de bit actual puede seguir siendo usada. Esto se ilustra también, por ejemplo, en la figura 4, como el uso de dos bucles diferentes. La figura 4 muestra uno, un bucle interior de precisión baja, que se ejecuta 405 cuando se determina 403 que es suficiente con una longitud de palabra de bit actual (menor); y uno, un bucle interior de precisión alta, que se ejecuta cuando se determina 403 que se necesita una longitud de palabra de bit más alta para representar la energía en el bucle interior, a fin de no perder información.

El método construye sobre la realización que el aumento máximo posible de una variable energía, tal como $enloop_y$, en un siguiente bucle interior ocurrirá cuando el pulso unidad se añada a la posición en y asociada con la $maxamp_y$ actual. Habiendo realizado esto, es posible determinar, antes de entrar en el bucle interior, si hay un riesgo o no de exceder la capacidad de representación de la longitud de palabra de bit usada actualmente, por ejemplo, 16 bits, durante el siguiente bucle interior. En otras palabras, la determinación de si se necesita más de una longitud de palabra de bit actual para representar $enloop_y$ comprende determinar las características del caso cuando, en el bucle de búsqueda interior próximo, el pulso unidad se añade a la posición en y que está asociada con $maxamp_y$. Por ejemplo, el número de bits necesario para representar $enloop_y$ en el bucle interior próximo se puede determinar o, alternativamente, un margen restante en una palabra de bit que representa $enloop_y$ en el bucle interior próximo.

Para vectores de forma objetivo que están asociados con un K bajo, es posible decir por adelantado que no habrá ninguna necesidad de una longitud de palabra de bit más larga que la ofrecida por la longitud de palabra de bit usada inicial o actualmente. Por lo tanto, sería posible aplicar un valor umbral Tk, de manera que se realicen ciertas operaciones solamente para vectores de forma objetivo que están asociados con un K que excede el valor umbral Tk. Para tales vectores objetivo, el codificador hará el seguimiento de $maxamp_y$ actualizando este valor después de cada adición de pulso. Para vectores objetivo asociados con un K que es menor que el valor umbral, no es necesario hacer el seguimiento de $maxamp_y$. Para el ejemplo con palabras de 16 y 32 bits, un posible Tk sería 127, como se describió previamente. En otras palabras, la realización del seguimiento de $maxamp_y$ y la determinación de si se necesita más de una longitud de palabra de bit actual se realiza, por ejemplo, solamente cuando el valor final de K asociado con el vector de forma objetivo de entrada excede un valor umbral Tk.

Una realización ilustrada en la figura 2 comprende hacer el seguimiento de o determinar 201 $maxamp_y$ y determinar 202 xabsmax. El valor de $maxamp_y$ se puede cambiar cuando se añade un nuevo pulso unidad en el bucle interior y por lo tanto $maxamp_y$ necesita ser actualizado, a fin de ser mantenido actualizado después de cada bucle. Por ejemplo, la acción 201 puede comprender hacer el seguimiento de $maxamp_y$ hasta que un valor de k haya alcanzado un valor umbral donde la longitud de palabra de bit inicial o actual usada para representar $enloop_y$ puede no ser ya suficiente y se comienza el análisis representado por ejemplo por la acción 204. La actualización de $maxamp_y$ después de un bucle interior que sigue al análisis de, por ejemplo, la acción 204 se ilustra como la acción 206 en la figura 2. Se debería señalar, no obstante, que xabsmax no se cambia en el proceso y, por lo tanto, solamente necesita ser determinada 202 una vez. Como se ilustra en la figura 2, una realización del método también podría comprender, antes de entrar en 205 un siguiente bucle de dimensión interior para adición de pulso unidad, determinando 203, en base a un valor absoluto máximo, $xabs_{max}$, del vector de forma de entrada, x, un cambio ascendente posible, en una palabra de bit, del valor de correlación dentro del bucle del siguiente bucle, $corr_{xy}$, entre x y el vector y. El cambio ascendente también podría indicar un escalado ascendente. La ecuación 19 anterior ilustra la determinación del cambio ascendente máximo posible. Realizando esto, se puede asegurar que se mantienen tantos bits de información de correlación como sean posibles durante la evaluación de bucle interior, especialmente los más significativos. Se debería señalar aquí que el valor de correlación $corr_{xy}$, en forma de $corr_{xy}^2$, no necesita ser representado necesariamente de una manera sin pérdidas. La determinación del cambio ascendente máximo se puede realizar en una palabra de bit "más larga", con independencia de la longitud de palabra de bit actual usada en el bucle interior. Es decir, el cambio ascendente máximo posible se puede determinar para una palabra de 32 bits incluso cuando una palabra de 16 bits se use en el bucle interior. Cuando va a ser usada una palabra de bit más corta en el bucle interior, el cambio ascendente determinado entonces se "redondeará" a la palabra de bit más corta, como se ilustra por la Ec. 20. Señalar que el valor de correlación, $corr_{xy}$, siempre es menor o igual que uno (1,0) en la precisión del DSP aplicada para el valor de correlación, es decir, $corr_{xy} \leq 1{,}0$ y, por lo tanto, el cambio ascendente máximo determinado para $corr_{xy}$ también es válido para $corr_{xy}^2$.

Cuando se determina que se necesita más de una longitud de palabra de bit actual para representar $enloop_y$, los cálculos de bucle interior se pueden realizar usando una longitud de palabra de bit más larga (que la longitud de palabra de bit actual, por ejemplo, 32 en lugar de 16 bits) para representar $enloop_y$.

En una realización, cuando se determina que se necesita más de una longitud de palabra de bit actual para representar *enloop*$_y$, los cálculos de bucle interior se realizan con una longitud de palabra de bit más larga (que la longitud de palabra de bit actual), que representa también un valor de correlación dentro del bucle acumulado, $corr_{xy}^2$, en el bucle interior. Esto se ilustra por ejemplo en la figura 3, en la acción 305. Es decir, una longitud de palabra de bit determinada para el valor de energía *enloop*$_y$ también se puede aplicar para $corr_{xy}^2$.

Como se mencionó previamente, se prefiere evitar realizar la división de la Ec. 8 en el bucle de búsqueda de dimensión interior para la adición de pulso unidad. Por lo tanto, se puede realizar una multiplicación cruzada, como se ilustra en la Ec. 10. Es decir, una posición n_{best} en y para adición de un pulso unidad, se podría determinar evaluando una multiplicación cruzada, para cada posición n en y, de un valor de correlación y de energía para la n actual; y una correlación "mejor hasta ahora", BestCorrSq y un valor de energía "mejor hasta ahora" bestEn, guardados a partir de valores previos de n, como:

$$corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

donde

$$\left.\begin{aligned} n_{best} &= n \\ bestEn &= enloop_y \\ BestCorrSq &= corr_{xy}^2 \end{aligned}\right\} \text{cuando } corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

La posición n_{best} podría ser referida como una posición "mejor" en y para adición de un pulso unidad. Se debería señalar que "$\geq$" se podría usar en las expresiones anteriores en lugar de ">". No obstante, ">", es decir "mayor que" se puede preferir cuando se intenta mantener el coste de cálculo tan bajo como sea posible, por ejemplo, con respecto a un número de ciclos.

La realización del método según cualquiera de las realizaciones descritas anteriormente permite que esta multiplicación cruzada sea realizada de una manera eficiente (por ejemplo, no usando una precisión más alta que la necesaria realmente).

Implementaciones

Los métodos y técnicas descritas anteriormente se pueden implementar en un codificador o códec, que puede estar comprendido por ejemplo en un dispositivo de comunicación.

Codificador, figuras 11a-11c

Una realización ejemplar de un codificador se ilustra de una manera general en la figura 11a. El codificador puede ser un codificador de medios, configurado para codificación de, por ejemplo, señales de audio y/o vídeo. El codificador 1100 se configura para realizar al menos una de las realizaciones del método descritas anteriormente con referencia a cualquiera de las figuras 1-5. El codificador 1100 está asociado con los mismos rasgos técnicos, objetos y ventajas que las realizaciones del método descritas previamente. En algunas implementaciones, el codificador está asociado con restricciones con respecto a memoria y/o complejidad, tales como, por ejemplo, cuando el codificador se configura con un DSP de precisión fija. El codificador se describirá brevemente a fin de evitar una repetición necesaria.

El codificador se puede implementar y/o describir como sigue:

El codificador 1100 se configura para una Cuantificación de Vector en Pirámide, incluyendo la denominada búsqueda fina o búsqueda de forma fina, donde un Cuantificador de Vector en Pirámide, PVQ, se configura para tomar un vector objetivo x como entrada y deriva un vector y añadiendo iterativamente pulsos unidad en un bucle de búsqueda de dimensión interior. El vector de entrada x tiene una dimensión N y una norma L1 de K. El codificador 1100 comprende circuitería de procesamiento o medios de procesamiento 1101 y una interfaz de comunicación 1102. La circuitería de procesamiento 1101 se configura para hacer al codificador 1100, antes de entrar en un siguiente bucle de búsqueda de dimensión interior para adición de un pulso unidad: determinar, en base a una amplitud de pulso máxima, *maxamp*$_y$, de un vector y actual, si se necesita más de una longitud de palabra de bit actual para representar, de una manera sin pérdidas, una variable, *enloop*$_y$, relacionada con una energía acumulada de y, en el bucle de dimensión interior próximo. La interfaz de comunicación 1102, que también se puede denotar por ejemplo interfaz de Entrada/Salida (I/O), incluye una interfaz para enviar datos a y recibir datos desde otras entidades o módulos.

La circuitería de procesamiento 1101 podría, como se ilustra en la figura 11b, comprender un medio de procesamiento, tal como un procesador 1103, por ejemplo, una CPU y una memoria 1104 para almacenar o contener las instrucciones. La memoria entonces comprendería instrucciones, por ejemplo, en forma de un programa

de ordenador 1105, que cuando se ejecutan por el medio de procesamiento 1103 hacen al codificador 1100 realizar las acciones descritas anteriormente.

Una implementación alternativa de la circuitería de procesamiento 1101 se muestra en la figura 11c. La circuitería de procesamiento aquí comprende una unidad de determinación 1106, configurada para hacer al codificador 1100, antes de entrar en un siguiente bucle de búsqueda de dimensión interior para adición de un pulso unidad: determinar, en base a una amplitud de pulso máxima, $maxamp_y$, de un vector y actual, si se necesita una precisión más alta que la que se permite con una longitud de palabra de bit actual para representar, de una manera sin pérdidas, una variable, $enloop_y$, relacionada con una energía acumulada de y, en el bucle de dimensión interior próximo. La circuitería de procesamiento 1101 podría comprender más unidades, tales como una unidad de búsqueda fina 1107, configurada para hacer al codificador ejecutar un bucle de dimensión interior con una cierta longitud de palabra de bit y/o una cierta precisión.

Los codificadores descritos anteriormente se podrían configurar para las diferentes realizaciones del método descritas en la presente memoria, tales como, por ejemplo, realizar los cálculos de bucle interior usando una palabra de bit más larga que representa $enloop_y$ y posiblemente $corr_{xy}^2$, cuando se determina que más de una longitud de palabra de bit actual es necesaria para representar $enloop_y$. “Más larga”, aquí se refiere a más larga que una longitud de palabra de bit actual o inicial.

El codificador 1100 se puede suponer que comprende funcionalidad adicional, para llevar a cabo funciones habituales de codificador.

El codificador descrito anteriormente puede estar comprendido en un dispositivo, tal como un dispositivo de comunicación. El dispositivo de comunicación puede ser un equipo de usuario (UE) en forma de un teléfono móvil, cámara de vídeo, grabadora de sonido, tableta, ordenador de sobremesa, ordenador portátil, receptor multimedia digital de TV o servidor doméstico/pasarela doméstica/punto de acceso doméstico/encaminador doméstico. El dispositivo de comunicación puede ser, en algunas realizaciones, un dispositivo de red de comunicaciones adaptado para codificación y/o transcodificación. Ejemplos de tales dispositivos de red de comunicaciones son servidores, tales como servidores de medios, servidores de aplicaciones, encaminadores, pasarelas y estaciones base radio. El dispositivo de comunicación también se puede adaptar para ser colocado en, es decir, que se integre en, un buque, tal como un barco, avión radiocontrolado, avión o un vehículo de carretera, tal como un coche, autobús o camión. Tal dispositivo integrado pertenecería típicamente a una unidad telemática del vehículo o sistema de información y entretenimiento del vehículo.

Los pasos, funciones, procedimientos, módulos, unidades y/o bloques descritos en la presente memoria se pueden implementar en hardware usando cualquier tecnología convencional, tal como tecnología de circuitos discretos o circuitos integrados, incluyendo tanto circuitería electrónica de propósito general como circuitería de aplicaciones específicas.

Ejemplos particulares incluyen uno o más procesadores de señal digital configurados adecuadamente y otros circuitos electrónicos conocidos, por ejemplo, puertas lógicas discretas interconectadas para realizar una función especializada o Circuitos Integrados de Aplicaciones Específicas (ASIC).

Alternativamente, al menos algunos de los pasos, funciones, procedimientos, módulos, unidades y/o bloques descritos anteriormente se pueden implementar en software tal como un programa de ordenador para ejecución por circuitería de procesamiento adecuada incluyendo una o más unidades de procesamiento. El software se podría transportar por un portador, tal como una señal electrónica, una señal óptica, una señal radio o un medio de almacenamiento legible por ordenador antes y/o durante el uso del programa de ordenador en el dispositivo de comunicación.

El diagrama o diagramas de flujo presentados en la presente memoria se pueden considerar como un diagrama o diagramas de flujo de ordenador, cuando se realizan por uno o más procesadores. Un aparato correspondiente se puede definir por un grupo de módulos de función, donde cada paso realizado por el procesador corresponde a un módulo de función. En este caso, los módulos de función se implementan como un programa de ordenador que se ejecuta en el procesador. Se tiene que entender que los módulos de función no tienen que corresponder a módulos software reales.

Ejemplos de circuitería de procesamiento incluyen, pero no se limitan a, uno o más microprocesadores, uno o más Procesadores de Señal Digitales, DSP, una o más Unidades Centrales de Proceso, CPU y/o cualquier circuitería de lógica programable adecuada tal como una o más Disposiciones de Puertas Programables en Campo, FPGA o uno o más Controladores Lógicos Programables, PLC. Es decir, las unidades o módulos en las disposiciones en los diferentes dispositivos descritos anteriormente se podrían implementar por una combinación de circuitos analógicos y digitales y/o uno o más procesadores configurados con software y/o microprogramas, por ejemplo, almacenados en una memoria. Uno o más de estos procesadores, así como el otro hardware digital, se puede incluir en una única circuitería integrada de aplicaciones específicas, ASIC o varios procesadores y diverso hardware digital se puede distribuir entre varios componentes separados, ya sea empaquetados o ensamblados individualmente en un sistema en un chip, SoC.

También se debería entender que puede ser posible reutilizar las capacidades generales de procesamiento de cualquier dispositivo o unidad convencional en el que se implementa la tecnología propuesta. También puede ser posible reutilizar software existente, por ejemplo, mediante la reprogramación de software existente o añadiendo nuevos componentes software.

Realizaciones ejemplares adicionales

Expresado de una manera ligeramente diferente, la descripción en la presente memoria se refiere a, por ejemplo, los siguientes aspectos y realizaciones.

Uno de los aspectos es un codificador/códec, en donde el codificador/códec se configura para realizar uno, más de uno o incluso todos los siguientes pasos, ilustrados, por ejemplo, en las figuras 5-6:

- determinar, calcular u obtener un valor absoluto máximo ($xabs_{max}$) de un vector objetivo de entrada (x(n)), por ejemplo, según las ecuaciones 11 y 12 anteriores y como se ilustra, por ejemplo, con el paso S1 en la figura 5 en una realización,

- determinar, calcular u obtener un cambio ascendente posible de un valor de correlación en base al menos al valor absoluto máximo ($xabs_{max}$), por ejemplo, calculando el cambio ascendente posible de un valor de correlación dentro del bucle acumulado del siguiente bucle en una palabra de 32 bits con signo a través de la ecuación 19 anterior e ilustrado con el paso S2 en la figura 5 en una realización,

- si el número de pulsos unidad finales (K) terminase siendo más alto que un umbral (*tp*), el cual, por ejemplo, puede ser 127 pulsos unidad, determinar, por ejemplo, hacer el seguimiento de/almacenar, un valor/información de amplitud de pulso máxima ($maxamp_y$) calculado, por ejemplo, según la ecuación 26 anterior de un vector (y(n)), que se puede definir según las ecuaciones 13 y 14 anteriores y

  - determinar/calcular/decidir/seleccionar en base a la amplitud de pulso máxima almacenada, por ejemplo, a través de un cálculo según las ecuaciones 22 y 23 de más adelante y como se ilustra por el paso S3 en la figura 6, si se necesita o se debería usar más de una cierta longitud de palabra, por ejemplo, más de una palabra de 16 bits con signo o más de una palabra de 32 bits con signo para representar energía dentro del bucle sin perder o sustancialmente perder, cualquier información de energía,

  - representar un término/parámetro/valor de correlación cuadrada mejor y un término/parámetro/valor de energía acumulada mejor mediante más de la cierta longitud de palabra, por ejemplo, palabras de 32 bits o palabras de 64 bits, si se necesita más de la cierta longitud de palabra y

  - si se necesita menos de la cierta longitud de palabra, ejecutar un primer bucle,

  - si se necesita más de la cierta longitud de palabra, ejecutar un segundo bucle, alternativo, con el término de energía acumulada “mejor hasta ahora” (casi óptima) y el término de correlación cuadrada mejor representados por las más de las palabras de cierta longitud de palabra.

El segundo bucle puede ser un bucle de pulso unidad de mayor precisión y alta precisión más intensivo computacionalmente que el primer bucle de menor precisión (es decir, en relación con el segundo bucle). La selección basada en energía acumulada dentro del bucle de la precisión de bucle interior tiene el efecto de que los subvectores objetivo que tienen picos altos o tienen una granularidad muy fina, (*K* final es alto) usarán o podría ser que usen el bucle de precisión más alta y más ciclos más a menudo, mientras que los subvectores de granularidad de pulsos sin picos o baja usarán o podrían usar más a menudo el bucle de precisión menor y menos ciclos.

Un aspecto se refiere a un dispositivo de comunicación 1, ilustrado en la figura 9, que comprende un codificador o códec 2 para codificación de vídeo o audio, por ejemplo, un codificador EVS.

El codificador o códec se puede implementar total o parcialmente como un DSP colocado en el dispositivo de comunicación. En una primera realización el codificador/códec se configura para hacer una búsqueda de forma de PVQ en base a un subvector objetivo (x(n)), el número de pulsos unidad finito (K), un valor de dimensión de subvector (N) del subvector objetivo y opcionalmente también uno o más valores de ganancia (g_{sub}). El codificador o códec también se puede configurar para hacer una división de banda de PVQ y en tal caso la búsqueda de forma de PVQ también se basaría en un número/valor de subvectores de una banda (N_S) y una ganancia más grande de un vector de ganancia *G*, (g_{max} = max (*G*) = max ($g_0 \ldots g_{(Ns-1)}$). El codificador o códec se configura además para sacar a partir de la búsqueda de forma de PVQ un vector entero (y) y/o un subvector de forma $x_q(n)$ a ser usado por el codificador para indexar un PVQ. El vector entero (y) comprende valores de elemento y tiene la misma longitud que el valor de dimensión de subvector (N) y una suma absoluta de todos los valores de elemento es igual al número de pulsos unidad (K).

El codificador/códec/dispositivo de comunicación se configura para realizar la búsqueda de forma de PVQ, en donde el codificador/códec/dispositivo de comunicación se configura para:

- determinar, calcular u obtener (S1, S23) un valor absoluto máximo ($xabs_{max}$) del vector de entrada (objetivo) (x(n)), por ejemplo, según las ecuaciones 11 y 12 anteriores,

- determinar, calcular u obtener (S2, S28) un cambio ascendente posible de un valor de correlación en base al menos al valor absoluto máximo ($xabs_{max}$), por ejemplo, calculando el cambio ascendente posible de un valor de correlación dentro del bucle acumulado del siguiente bucle en una palabra de 32 bits con signo a través de la ecuación 19 anterior,

- si el número de pulsos unidad finales (K) terminase siendo más alto que un umbral (*tp*), el cual, por ejemplo, puede ser 127 pulsos unidad, hacer el seguimiento de/almacenar (S30) un valor/información de amplitud de pulso máxima ($maxamp_y$) calculado, por ejemplo, según la ecuación 26 anterior de un vector (y(n)), que se puede definir según las ecuaciones 13 y 14 anteriores y

  - determinar/calcular/decidir/seleccionar (S3, S32) en base a la amplitud de pulso máxima almacenada, por ejemplo, a través de un cálculo según las ecuaciones 22 y 23 anteriores, si se necesita o se debería usar más de una cierta longitud de palabra, por ejemplo, más de una palabra de 16 bits con signo o más de una palabra de 32 bits con signo, para representar energía dentro del bucle,

  - representar (S34) un término/parámetro/valor de correlación cuadrada mejor y un término/parámetro/valor de energía acumulada mejor mediante más de la cierta longitud de palabra, por ejemplo, palabras de 32 bits o palabras de 64 bits, si se necesita más de la cierta longitud de palabra y

  - si se determina menos de la cierta longitud de palabra, ejecutar (S33) un primer bucle,

  - si se determina más de la cierta longitud de palabra, ejecutar (S35) un segundo bucle, alternativo, con el término de energía acumulada mejor y el término de correlación cuadrada mejor representados por las más de las palabras de cierta longitud de palabra.

La búsqueda de forma de PVQ anterior, que puede ser una búsqueda de forma de PVQ de precisión limitada, se realiza en una realización por un cuantificador de vector, el cual es parte del codificador/códec y se puede implementar, al menos parcialmente, pero también completamente como una unidad de DSP, que se puede colocar en o adaptar para ser colocada en un dispositivo de comunicación. De esta manera, el codificador/códec se puede implementar completamente o parcialmente como una unidad hardware, por ejemplo, un DSP o una disposición de puertas programables en campo (FPGA). Se puede implementar, no obstante, en realizaciones alternativas con la ayuda de un procesador de propósito general y un programa de ordenador de códec que cuando se ejecuta en el procesador de propósito general hace al dispositivo de comunicación realizar uno o más de los pasos mencionados en el párrafo anterior. El procesador también puede ser un procesador de Cálculo de Conjunto de Instrucciones Reducido (RISC).

Otro aspecto de la descripción en la presente memoria es, como se indicó en el párrafo anterior, un programa de ordenador 6 ilustrado en la figura 10 y donde se describe completamente una realización por el ejemplo de código ANSI-C en el apéndice 1 de más adelante, tal como un programa de ordenador de codificador o programa de ordenador de códec, que comprende un código legible por ordenador, que cuando se ejecuta en un procesador/unidad de procesador 4 de un dispositivo de comunicación 1 hace al dispositivo de comunicación realizar uno o más de los pasos mencionados en conjunto con el método en el párrafo de más adelante o cualquiera de los pasos mencionados en conjunto con la figura 7.

Aún otro aspecto es un método de búsqueda de forma de PVQ realizado por un dispositivo de comunicación/códec/codificador, en donde el método comprende uno o más de los siguientes pasos:

- determinar, calcular u obtener (S1) un valor absoluto máximo ($xabs_{max}$) del vector de entrada (objetivo) (x(n)), por ejemplo, según las ecuaciones 11 y 12 anteriores,

- determinar, calcular u obtener (S2, S28) un cambio ascendente posible de un valor de correlación en base al menos al valor absoluto máximo ($xabs_{max}$), por ejemplo, calculando el cambio ascendente posible de un valor de correlación dentro del bucle acumulado del siguiente bucle en una palabra de 32 bits con signo a través de la ecuación 19 anterior,

- si el número de pulsos unidad finales (K) terminase siendo más alto que un umbral (*tp*), el cual, por ejemplo, puede ser 127 pulsos unidad, hacer el seguimiento de/almacenar un valor/información de amplitud de pulso máxima ($maxamp_y$) calculado, por ejemplo, según la ecuación 26 anterior de un vector (y(n)), que se puede definir según las ecuaciones 13 y 14 anteriores y

  - determinar/calcular/decidir/seleccionar (S3) en base a la amplitud de pulso máxima almacenada, por ejemplo, a través de un cálculo según las ecuaciones 22 y 23 anteriores, si se necesita o se debería usar más de una cierta longitud de palabra, por ejemplo, más de una palabra de 16 bits con signo o más de una palabra de 32 bits con signo, para representar energía dentro del bucle,

- representar un término/parámetro/valor de correlación cuadrada mejor y un término/parámetro/valor de energía acumulada mejor mediante más de la cierta longitud de palabra, por ejemplo, palabras de 32 bits o palabras de 64 bits, si se necesita más de la cierta longitud de palabra y

- si se determina menos de la cierta longitud de palabra, ejecutar un primer bucle,

- si se determina más de la cierta longitud de palabra, ejecutar un segundo bucle, alternativo, con el término de energía acumulada mejor y el término de correlación cuadrada mejor representados por las más de las palabras de cierta longitud de palabra.

El dispositivo de comunicación puede ser un equipo de usuario (UE) en forma de un teléfono móvil, cámara de vídeo, grabadora de sonido, tableta, ordenador de sobremesa, ordenador portátil, receptor multimedia digital de TV o servidor doméstico/pasarela doméstica/punto de acceso doméstico/encaminador doméstico, etc. como se definió anteriormente.

Aún otro aspecto es un medio de almacenamiento legible por ordenador 5 (ver la figura 10) en el cual se almacena cualquiera de las realizaciones anteriores del programa de ordenador. El medio de almacenamiento legible por ordenador puede ser en forma de una memoria volátil o no volátil, por ejemplo una EEPROM (PROM Borrable Eléctricamente), FPGA, una memoria instantánea (incluyendo una unidad de estado sólido) y un disco duro.

Una realización del dispositivo de comunicación 1 se ilustra en la figura 9. El dispositivo de comunicación comprende, para la realización de una búsqueda de forma de PVQ, una o más de una o todas de las siguientes unidades:

- una primera unidad de determinación, U1, para determinar, calcular u obtener un valor absoluto máximo ($xabs_{max}$) del vector de entrada (objetivo) (x(n)), por ejemplo, según las ecuaciones 11 y 12 anteriores,

- una segunda unidad de determinación, U2, para determinar, calcular u obtener un cambio ascendente posible de un valor de correlación en base al menos al valor absoluto máximo ($xabs_{max}$), por ejemplo, calculando el cambio ascendente posible de un valor de correlación dentro del bucle acumulado del siguiente bucle en una palabra de 32 bits con signo a través de la ecuación 19 anterior,

- una unidad de almacenamiento, U3, para hacer el seguimiento de/almacenar un valor/información de amplitud de pulso máxima ($maxamp_y$) calculado, por ejemplo, según la ecuación 26 anterior de un vector (y(n)), que se puede definir según las ecuaciones 13 y 14 anteriores, si el número de pulsos unidad finales (K) terminase siendo más alto que un umbral (*tp*),

- una unidad de selección, U4, para determinar/calcular/decidir/seleccionar en base a la amplitud de pulso máxima almacenada, por ejemplo, a través de un cálculo según las ecuaciones 22 y 23 anteriores, si se necesita o se debería usar más de una cierta longitud de palabra, por ejemplo, más de una palabra de 16 bits con signo o más de una palabra de 32 bits con signo, para representar energía dentro del bucle,

- una unidad de representación, U5, para generar un término/parámetro/valor de correlación cuadrada mejor y un término/parámetro/valor de energía acumulada mejor con una longitud de palabra, por ejemplo, palabras de 32 bits o palabras de 64 bits, que son más de la cierta longitud de palabra si se selecciona por la unidad de selección más de la cierta longitud de palabra y

- una unidad de procesamiento de bucle interior, U6, para

- ejecutar un primer bucle, si se selecciona por la unidad de selección menos de la cierta longitud de palabra y

- ejecutar un segundo bucle, alternativo, con el término de energía acumulada mejor y el término de correlación cuadrada mejor representados por las más de las palabras de cierta longitud de palabra, si se determina más de la cierta longitud de palabra.

Las unidades mencionadas en el párrafo anterior pueden estar comprendidas en un códec/codificador 2 en forma de un DSP en la unidad de comunicación y pueden estar comprendidas además en un cuantificador de vector hardware del DSP. En una realización alternativa, todas las unidades en el párrafo anterior se implementan en el dispositivo de comunicación como software.

Como se ilustra además en la figura 9, el dispositivo de comunicación 1 también puede comprender unidades adicionales relacionadas con el codificador/códec y en particular unidades relacionadas con cuantificación de vector y búsqueda de forma de PVQ. Tales unidades se configuran para permitir búsquedas de forma según la descripción y figuras comprendidas en esta solicitud. Las unidades ejemplares ilustradas en la figura 9 son:

- una unidad de división de banda de PVQ U7 para realizar el paso opcional S21 descrito en conjunto con la figura 7,

- una unidad de comparación U8 para realizar el paso S24 descrito en conjunto con la figura 7 de más adelante,

- una unidad de generación de vector de PVQ U9 para realizar el paso S25 descrito más adelante,

- una unidad de generación de punto de inicio U10 para realizar el paso S26 descrito más adelante,

- una unidad de cálculo de parámetro U11 para realizar el paso S27 descrito más adelante,

- una unidad de asignación de bit U12 para, por ejemplo, suministrar K y N a la búsqueda de forma y

- una unidad de indexación de PVQ U13, que se puede ver como un receptor de la salida a partir de la búsqueda de forma de PVQ descrita en la presente memoria,

- una unidad de normalización U14 para realizar el paso S36 descrito más adelante y

- una unidad de salida U15 para realizar el paso S37 descrito más adelante.

En el caso de una implementación software en un dispositivo de comunicación, una realización del dispositivo de comunicación 1 se puede definir como un dispositivo de comunicación que comprende un procesador 4 y un producto de almacenamiento de programa de ordenador 5 en forma de una memoria, dicha memoria que contiene instrucciones ejecutables por dicho procesador, por lo cual dicho dispositivo de comunicación es operativo para realizar una, más de una o todas de las siguientes:

- determinar, calcular u obtener un valor absoluto máximo ($xabs_{max}$) del vector de entrada (objetivo) (x(n)), por ejemplo, según las ecuaciones 11 y 12 anteriores,

- determinar, calcular u obtener un cambio ascendente posible de un valor de correlación en base al menos al valor absoluto máximo ($xabs_{max}$), por ejemplo, calculando el cambio ascendente posible de un valor de correlación dentro del bucle acumulado del siguiente bucle en una palabra de 32 bits con signo a través de la ecuación 19 anterior,

- si el número de pulsos unidad finales (K) terminase siendo más alto que un umbral (*tp*), el cual, por ejemplo, puede ser 127 pulsos unidad, hacer el seguimiento de/almacenar un valor/información de amplitud de pulso máxima ($maxamp_y$) calculado, por ejemplo, según la ecuación 26 anterior de un vector (y(n)), que se puede definir según las ecuaciones 13 y 14 anteriores y

  - determinar/calcular/decidir/seleccionar en base a la amplitud de pulso máxima almacenada, por ejemplo, a través de un cálculo según las ecuaciones 22 y 23 anteriores, si se necesita o se debería usar más de una cierta longitud de palabra, por ejemplo, más de una palabra de 16 bits con signo o más de una palabra de 32 bits con signo, para representar energía dentro del bucle,

  - representar un término/parámetro/valor de correlación cuadrada mejor y un término/parámetro/valor de energía acumulada mejor mediante más de la cierta longitud de palabra, por ejemplo, palabras de 32 bits o palabras de 64 bits, si se necesita más de la cierta longitud de palabra y

  - si se determina menos de la cierta longitud de palabra, ejecutar un primer bucle,

  - si se determina más de la cierta longitud de palabra, ejecutar un segundo bucle, alternativo, con el término de energía acumulada mejor y el término de correlación cuadrada mejor representados por las más de las palabras de cierta longitud de palabra.

Para ilustrar aspectos y realizaciones adicionales, algunos de ellos están yendo a ser descritos a continuación en conjunto con las figuras 7-8.

La figura 8 proporciona una descripción general de un lado de transmisión del EVS del 3GPP emergente, incluyendo un codificador EVS 3, que aquí está comprendido en el dispositivo de comunicación 1.

La figura 7 ilustra algunos pasos del método en una forma alternativa de descripción de algunas realizaciones, en relación a las realizaciones ilustradas en las figuras 5-6. Incluso aunque algunos de los pasos mencionados con respecto a la figura 7 se puede decir que se hacen en conjunto con una búsqueda de forma de PVQ, también debería ser evidente que algunos de los pasos también se podría decir que se realizan antes de la búsqueda de forma de PVQ. En un primer paso opcional S21, se realiza una división de banda de PVQ.

Los subvectores objetivo de forma, opcionalmente desde el paso S21, se reciben en un segundo paso S22, en donde, en dependencia de la realización, también se pueden recibir g_{sub}, g_{max} y N_s.

En un tercer paso S23, que corresponde al paso S1 en la figura 5, se determina un valor absoluto máximo de un vector objetivo, por ejemplo, calculando primero el valor absoluto del subvector x(n) del vector objetivo y entonces seleccionando el valor absoluto más grande del subvector.

En un cuarto paso opcional S24, se determina si el valor del vector objetivo es igual a o está por debajo de un primer umbral. El umbral se fija para “filtrar” los vectores objetivo que se considera que tienen valores de energía muy bajos. Como se explicó anteriormente, el umbral se podría fijar para ser igual a cero en una realización. Se podría

decidir también en este cuarto paso si una ganancia de subvector es igual a o está por debajo de un segundo umbral. En una realización el segundo umbral se fija a cero, pero en otras realizaciones se puede fijar para ser la Épsilon de la Máquina en dependencia de la precisión usada para palabras procesadas.

Si se determina en el cuarto paso S24 que el vector objetivo es igual o está por debajo del primer umbral y/o la ganancia de subvector está por debajo o es igual al segundo umbral, entonces se crea un vector de PVQ en un quinto paso S25 opcional. La creación se crea en una realización de manera determinística asignando la mitad de los K pulsos unidad a una primera posición ($y[0]=\left\lfloor \frac{K}{2} \right\rfloor$) y los pulsos unidad restantes a una última posición (y[N-1]=y[N-1]+(K-y[0])). Este paso se podría ver en conjunto con el cuarto paso S24 como que deja de lado la búsqueda de forma de PVQ real entera, pero también se puede ver como una subrutina dentro del contexto de un procedimiento de búsqueda de forma de PVQ general.

En un sexto paso S26 opcional, un valor inicial (punto de inicio) para y, y_start se fija para la búsqueda de forma de PVQ para seguir, en donde el valor inicial es dependiente de la relación entre K y N. Si la relación es mayor que un tercer valor umbral, el cual puede ser 0,5 pulsos unidad por coeficiente, una primera proyección a una subpirámide K-1 se usa como el vector inicial y_start en un siguiente paso. La primera proyección se puede calcular como en las ecuaciones 13 y 14 anteriores. Si es menor que el tercer umbral, entonces el vector inicial y_start se decide que parta desde 0 pulsos unidad colocados previamente.

En la preparación para los pasos de búsqueda de forma de PVQ posteriores, todos los valores de vector iniciales en y_start se fijan a cero en un séptimo paso S27. En este paso se calculan un primer parámetro, aquí llamado el número acumulado de pulsos unidad, $pulse_{tot}$ y un segundo parámetro, aquí la correlación acumulada, $corr_{xy}(pulse_{tot})$ y un tercer parámetro, aquí llamado la energía acumulada $energy_y(pulse_{tot})$ para el punto de inicio, por ejemplo, según las ecuaciones 15-17 respectivamente. Un cuarto parámetro, aquí llamado $enloop_y(pulse_{tot})$ se puede calcular también en este paso según la ecuación 18 anterior.

En un octavo paso S28, se inicia una búsqueda de forma de PVQ o en una forma alternativa de mirarla, la segunda parte de búsqueda, fina, de la búsqueda de forma de PVQ se inicia para los pulsos unidad restantes hasta K con la ayuda de K, N, X_abs, max_xabs e y obtenidos, determinados o calculados previamente y en algunas realizaciones también g_{sub}, g_{max} y N_S. Los pasos detallados de algunas realizaciones de esta búsqueda fina se ilustran minuciosamente por ejemplo por la figura 6, pero se podría enfatizar que en algunas realizaciones la búsqueda fina comprende una determinación de un quinto parámetro/valor, aquí llamado un cambio ascendente de un valor de correlación, $corr_{upshift}$, se calcula para al menos algunos y en algunas realizaciones todos, los pulsos unidad para los cuales se hace la búsqueda fina o bucle interior. En algunas realizaciones un cambio ascendente posible de un valor de correlación dentro del bucle acumulado del siguiente bucle en una palabra de 32 bits con signo se calcula en base a la ecuación 19 anterior y $corr_{upshift}$ se usa entonces como entrada a un cálculo de un valor de correlación $corr_{xy}$ en la ecuación 20.

En un noveno paso S29, que se puede decir que es una parte de la búsqueda de forma de PVQ fina, se determina si el número de pulsos unidad finales K terminaría siendo más alto que un tercer umbral, t_p, para el número de pulsos unidad finales. Si este es el caso, entonces en un décimo paso S30, se almacena la amplitud de pulso máxima $maxamp_y$.

En un undécimo paso S31, un sexto parámetro en_{margin} se calcula según, por ejemplo, la ecuación 22.

En un duodécimo paso S32, el sexto parámetro se compara con el cuarto valor umbral, que corresponde a una cierta longitud de palabra.

Si la respuesta es SÍ (en S32 de la Fig. 7) o Falso (en el ejemplo de código ANSI en el Apéndice 1), es decir, en_{margin} en la ecuación/decisión ejemplar 23 es mayor o igual que el cuarto umbral “16”, entonces en un paso S33, un primer bucle, más rápido y “más tosco” se ejecuta en un segundo bucle de la búsqueda fina. Las realizaciones del primer bucle se muestran, por ejemplo, en la figura 6.

Si la respuesta es “No” (en S32 de la Fig. 7) o Verdadero (en el ejemplo de código ANSI en el Apéndice 1), es decir, en_{margin} en la ecuación/decisión ejemplar 23 es menor que “16”, entonces en un decimocuarto paso S34, un séptimo parámetro, el término/parámetro/valor de correlación cuadrada mejor y un octavo parámetro, el término/parámetro/valor de energía acumulada mejor, se generan/transforman para llegar a ser más de la cierta longitud de palabra, que entonces en un decimoquinto paso S35 se usan en el segundo bucle interior, más detallado, de la búsqueda fina. Las realizaciones del segundo bucle se muestran en más detalle por ejemplo en la figura 6.

En un decimosexto paso S36, se asigna su signo adecuado a al menos cada elemento de subvector de PVQ no cero y el vector se normaliza L2 a energía unidad. Si, en algunas realizaciones, se ha dividido una banda, entonces se escala con una ganancia de subvector g_{sub}.

Un x_q normalizado también se puede determinar en base a la ecuación 28. Un procedimiento ejemplar para este paso se describió anteriormente más minuciosamente.

En un decimoséptimo paso S37, el x_q normalizado e y se sacan a partir del proceso de búsqueda de forma de PVQ y se reenvían a un proceso de indexación de PVQ incluido por ejemplo en el códec.

Algunas ventajas de realizaciones y aspectos

Más adelante están algunas ventajas sobre la técnica anterior habilitadas en al menos algunos de los aspectos y realizaciones descritas anteriormente.

El método/algoritmo de escalado de correlación propuesto que usa un análisis previo de la correlación máxima acumulada actual, mejora el rendimiento de la SNR del peor caso (mínima) de una implementación de búsqueda de cuantificación de forma de PVQ de precisión. El criterio adaptativo para análisis de margen de correlación por adelantado requiere una complejidad adicional muy marginal. Además se requiere una normalización previa no costosa del vector objetivo x por ejemplo a energía unidad.

El criterio adaptativo que usa seguimiento de la amplitud de pulso máxima en el resultado preliminar, seguido por un análisis previo de la energía acumulada del peor caso, por ejemplo para la decisión de bucle interior de precisión de 16/32 bits suave requiere muy poca complejidad de cálculo adicional y proporciona un buen compromiso donde la complejidad se puede mantener baja mientras que aún se usan métricas de correlación de alta precisión y de energía de alta precisión para señales de entrada relevantes y se asignará más precisión a señales de pico subjetivamente más importantes. En otras palabras, al menos algunas de las realizaciones y aspectos mejoran el funcionamiento de un ordenador/procesador en sí mismo.

En las Tablas 2/3 anteriores en el apéndice 2 más adelante, uno puede encontrar que un ejemplo de sistema basado en PVQ que usa coste lógico de precisión adaptativa será 6,843 WMOPS, si uno usase precisión de energía y de correlación cuadrada de 32 bits en todos (cualquier K) los bucles de búsqueda interior el coste se eleva a 10,474 WMOPS.

Observaciones finales

Las realizaciones descritas anteriormente se dan meramente como ejemplos y se debería entender que la tecnología propuesta no está limitada a las mismas. Se entenderá por los expertos en la técnica que se pueden hacer diversas modificaciones, combinaciones y cambios a las realizaciones sin apartarse del presente alcance. En particular, diferentes soluciones parciales en diferentes realizaciones se pueden combinar en otras configuraciones, donde sea técnicamente posible.

Cuando se usa la palabra “comprende” o “que comprende” se entenderá como no limitante, es decir, que significa “consta al menos de”.

También se debería señalar que en algunas implementaciones alternativas, las funciones/actos señalados en los bloques pueden ocurrir fuera del orden señalado en los diagramas de flujo. Por ejemplo, dos bloques mostrados en sucesión se pueden ejecutar de hecho sustancialmente concurrentemente o los bloques se pueden ejecutar algunas veces en el orden inverso, dependiendo de la funcionalidad/actos implicados. Además, la funcionalidad de un bloque dado de los diagramas de flujo y/o diagramas de bloques se puede separar en múltiples bloques y/o la funcionalidad de dos o más bloques de los diagramas de flujo y/o diagramas de bloques se puede integrar al menos parcialmente. Finalmente, se pueden añadir/insertar otros bloques entre los bloques que se ilustran y/o se pueden omitir bloques/operaciones sin apartarse del alcance de los conceptos inventivos.

Se tiene que entender que la elección de unidades de interacción, así como la denominación de las unidades dentro de esta descripción son solamente con propósito ejemplar y los nodos adecuados para ejecutar cualquiera de los métodos descritos anteriormente se pueden configurar de una pluralidad de formas alternativas a fin de ser capaces de ejecutar las acciones del procedimiento sugeridas.

También se debería señalar que las unidades descritas en esta descripción se tienen que considerar como entidades lógicas y no necesariamente como entidades físicas separadas.

Con referencia a un elemento en singular no se pretende que signifique “uno y solamente uno” a menos que se exponga así explícitamente, sino más bien “uno o más”. Todos los equivalentes estructurales y funcionales a los elementos de las realizaciones descritas anteriormente que son conocidos por los expertos en la técnica se incorporan expresamente en la presente memoria por referencia y se pretende que estén abarcados por este medio. Además, no es necesario para un dispositivo o método abordar todos y cada uno de los problemas que se busca sean resueltos por la tecnología descrita en la presente memoria, para ser abarcados por este medio.

En algunos casos en la presente memoria, se omiten descripciones detalladas de dispositivos, circuitos y métodos bien conocidos para no oscurecer la descripción de la tecnología descrita con detalle innecesario. Todas las declaraciones en la presente memoria que recitan principios, aspectos y realizaciones de la tecnología descrita, así

como ejemplos específicos de los mismos, se pretende que abarquen tanto equivalentes estructurales como funcionales de las mismos. Adicionalmente, se pretende que tales equivalentes incluyan tanto equivalentes conocidos actualmente así como equivalentes desarrollados en el futuro, por ejemplo, cualquier elemento desarrollado que realice la misma función, con independencia de la estructura.

Abreviaturas

N	dimensión de vector
N_s	dimensión de subvector
x	vector objetivo
x_q	vector de forma cuantificado
y_{final}	vector entero que se adhiere a la norma L1 de K
K	Número de pulsos unidad finales
k	número de índice de pulsos unidad acumulados
n	coeficiente o índice de muestra
i	índice de subvector
MDCT	Transformada Coseno Discreta Modificada
PVQ	Cuantificador de Vector en Pirámide (Cuantificación)
WC	Peor caso
WMOPS	Millón de Operaciones Ponderadas Por Segundo
AccEn	Energía Acumulada
ROM	Memoria de Sólo Lectura
PROM	ROM de Programa
SNR	Relación Señal a Ruido
EVS	Servicio de Voz Mejorado
3GPP	Proyecto de Cooperación de 3ª Generación
DSP	Procesador de Señal Digital
CELT	Transformada Repetida de Energía Limitada
IEFT	Grupo de Trabajo de Ingeniería de Internet
MAC	Multiplicar-Acumular
ACELP	Predicción lineal excitada por código algebraico
EPS	Épsilon de la máquina

Apéndice 1: Implementación ejemplar de realización en código ANSI-C

Más adelante está un ejemplo de una implementación de una realización ejemplar en código ANSI C que usa 16/32 bits virtuales de STL 2009 de la G.191 (una simulación de un DSP).

```
static

Word 16 max_val_fx(          /* o  : valor máximo en el vector de entrada                    */
        const Word16 *vec,   /* i  : vector de entrada                                       */
        const Word16 lvec    /* i : longitud de vector de entrada                            */
        )
{
  Word16 j,tmp;
  tmp = vec[0];        move16();
  FOR ( j=1 ; j<lvec ; j++ )
  {
    tmp = s_max(vec[j],tmp);
  }
  return tmp;

}
```

/* El bucle de búsqueda interior para un único pulso unidad adicional, que comienza desde pulse_tot, con información acerca de la precisión/reducción de energía requerida para el bucle tenue en en_dn_shift y el valor absoluto de max_xabs actual a ser usado para un cambio ascendente de correlación casi óptima, devuelve el índice del pulso unidad colocado mejor en imax

*/

```
static
Word16 one_pulse_search(const Word16 dim,          /* dimension de vector    */
        const Word16* x_abs,                 /* valores de vector absolutos */
        Word16* y,    /* vector de salida        */
        Word16 *pulse_tot_ptr,
        Word32* L_xy_ptr,        /* correlación acumulada */
        Word32* L_yy_ptr,        /* energía acumulada      */
        Word16  high_prec_active,
        Word16  en_dn_shift,
        Word16 max_xabs)       /* amplitud máxima acumulada actual para pulsos */

{
  Word16 i, corr_up_shift, corr_tmp, imax, corr_sq_tmp, en_max_den, cmax_num, en_tmp;
  Word32 L_tmp_en_lc, L_tmp_corr ;
  Word32 L_tmp_en,  L_en_max_den,  L_corr_sq_max,  L_tmp_corr_sq;
  Word32 L_left_h, L_right_h;
```

```
  UWord32 UL_left_l, UL_right_l, UL_dummy;
#ifndef NONBE_PVQ_SEARCH_FIX
  Word16 corr_margin;
#endif
#ifdef  BE_CLEAN_PVQ_SEARCH
  Word32  L_tmp;
  UWord32 UL_tmp;
  UWord16 u_sgn;

#endif

  /* maximizar la precisión de correlación, anterior a cada adición de pulso unidad en el vector */
  corr_up_shift      =  norm_l(L_mac(*L_xy_ptr,  1, max_xabs));
  /* analizar previamente la actualización de L_xy de peor caso en el bucle tenue            , 2 op. */
  /* borrar código BE, con bucles de precisión baja/alta divididos                                    */
  /* activar sección de búsqueda de en/corr de baja complejidad condicionalmente si la energía resultante está
  dentro de los límites */
  /* caso típico para dimensiones más altas                                                          */

  IF( high_prec_active == 0 )
  {
      en_max_den                        = 0; /*move16()*/; /* OPT: mover guardado usando high_prec_active como
                                                                en_max_den */
      cmax_num  = -1;  move16();   /* solicitar forzar una 1ª actualización para n==0  */
      FOR(i =0; i < dim; i++)            /* PARA 3 op.  */
      {
          L_tmp_corr                    = L_shl(L_mac(*L_xy_ptr, 1,x_abs[i]), corr_up_shift ); /*objetivo dentro del bucle
                                        real */
          corr_tmp                       = round_fx(L_tmp_corr);
          corr_sq_tmp                   = mult(corr_tmp, corr_tmp);  /* CorrSq, tiene 16bit para multiplicación cruzada de
                                        baja complejidad */


          L_tmp_en_lc                    = L_mac(*L_yy_ptr, 1,y[i] );
          /*resultado de Q1, la energía puede  abarcar hasta ~14+1(Q1)+1(signo)=16 bits, 1 op. */
          /* extract_l sin cambio se puede usar siempre para esta sección ya que se garantiza que permanece
            energía  en la palabra inferior, 1 op */
          en_tmp                      = extract_l(L_tmp_en_lc);
          /* L_shl + round_fx también se podrían usar pero entonces añade un coste de cambio ascendente (2-3
          op.)*/


          /* comparación de 16/32 bits     WC (4 +1+1 + (1+1+1) = 9                          */
          /*  corr_sq_tmp/en_tmp_den            >   cmax_num/en_max_den  */
          /*  corr_sq_tmp * en_max_den          >   cmax_num * en_tmp      */      /* SI       4 op  */
          IF( L_msu(L_mult(corr_sq_tmp,  en_max_den),cmax_num,  en_tmp)  >0)          /*        2 op  */
          {
            cmax_num     = corr_sq_tmp;  move16();   /* 1 op */
            en_max_den  = en_tmp;       move16();      /* 1 op */
```

```
            imax      = i;              move16();       /* 1 op. */
        }
    }/* tenue */
}
ELSE
{ /* Sección de alta resolución activada cuando la energía de vector está llegando a ser alta (con picos o muchos
pulsos) */
  /* Operador BASOP Mpy32_32_ss usado para permitir resolución más alta tanto para el término CorrSq
    como Energía */
  /* Rendimiento:   cercano a referencia flotante                                                    */




  L_en_max_den = L_deposit_l(0);                                                    /* 1 op */
  L_corr_sq_max = L_deposit_l(-1); /* solicitar forzar una 1ª actualización   */                    /* 1 op.  */

  FOR(i =0; i < dim; i++)                                                  /* PARA 3 op. */
  {
      L_tmp_corr                          = L_shl(L_mac(*L_xy_ptr, 1,x_abs[i]), corr_up_shift );/* valor de WC dentro
      del bucle real */
      Mpy_32_32_ss(L_tmp_corr,L_tmp_corr, &L_tmp_corr_sq, &UL_dummy);
      /* CorrSq 32 bits, 4 op. */

      L_tmp_en = L_mac(*L_yy_ptr, 1, y[i]);                          /* Q1,la energía puede extenderse hasta señalar
      +19 bits , 1 op.      */
      /* Para una exactitud de más alta usar pares de valores con signo de 32x32 bits cambiados ascendentemente
      máximos      */
      /*  (L_tmp_corr_sq / L_tmp_en)      >  (L_corr_sq_max/L_en_max_den)                                  */
      /*  (L_tmp_corr_sq * L_en_max_den)  >  (L_corr_sq_max * L_tmp_en)                                    */
      Mpy_32_32_ss( L_en_max_den, L_tmp_corr_sq, &L_left_h, &UL_left_l);                              /* 4 op. */
      Mpy_32_32_ss( L_tmp_en, L_corr_sq_max, &L_right_h, &UL_right_l);                               /* 4 op. */

      /* comparación de 32/64 bits WC (1+1+2x2+4 +(2+2+1) = 15                                          */
      L_tmp  = L_sub(L_left_h, L_right_h);                /* comprobar palabra con signo alta 1 op.  */

      UL_tmp = UL_subNs(UL_right_l, UL_left_l, &u_sgn);
      /* comprobar palabra con signo baja, señalar conmutador debido a ">=" en UL_subNs, 1 op. */
      IF( ( L_tmp >0)   ||   (L_tmp ==0) && (u_sgn !=0)))  /* SI 4 op. */
      {
        L_corr_sq_max =  L_tmp_corr_sq;  move32();  /* 2 op. */
        L_en_max_den  =  L_tmp_en;       move32();  /* 2 op. */
        imax          =  i;                  move16(); /* 1 op.  */
      }
    } /* bucle tenue */
  }
/* finalmente añadir contribución de pulso unidad encontrada a L_xy, Lyy pasados, para el siguiente bucle de pulso
      */
```

```
  *L_xy_ptr = L_mac(*L_xy_ptr, x_abs[imax], 1);      /* xy_flt += xabs[imax];   Q12+1 */
  *L_yy_ptr = L_mac(*L_yy_ptr,  1, y[imax]);          /* yy_flt += y[imax];       */

  y[imax]          = add(y[imax], 1);    move16();    /* pulso añadido de Q0       */
  (*pulse_tot_ptr)   = add((*pulse_tot_ptr)  ,1);   /* incremento de suma de pulsos total   */
  return imax;

}
```

```
/*-------------------------------------------------------------------------   *

* Función pvq_encode_fx()              *

*                                                                                  *
*  Algoritmo de búsqueda de PVQ básico:                                          *
*  Proyección de norma L1 selectiva a la pirámide ~(K-1) menor                    *
*  seguida por una búsqueda fina de adición de pulso unidad tipo ACELP            *
*-------------------------------------------------------------------------    */
void  pvq_encode_fx(
  Encoder_State_fx  *st_fx,
  const Word 16 *x,    /* i:   vector para cuantificar          Q15-3=>Q12     */
  Word16 *y,           /*o:   pulsos en bruto de Q0  (corto no escalado)     */
  Word16 *xq,          /*o:   vector cuantificado      Q15                    */

  const short pulses,          /* i:   número de pulsos asignados              */
  const short dim,             /* i:   Longitud de vector  == N                */
  const Word16 neg_gain   /* i:  -Ganancia de Subvector      usamos –ganancia negativa en  Q15  0..1   */
)
{
  Word16 i;
  Word16 pulse_tot;
  Word16 xabs[PVO_MAX_BAND _SIZE];
  Word16  max_xabs;
  Word32 L_xsum;
  Word32  L_proj_fac;
  Word32 L_yy, L_xy;
  Word16 max_amp_y, imax;
  Word16   k, en_margin, en_dn_shift, high_prec_active ;
  Word32 L_num, L_tmp;
  Word16  proj_fac, tmp, shift_den,shift_num,shift_delta,  num,den;
  UWord16 u16_tmp;
  Word16 dim_m1;
  Word32  L_isqrt;
  Word16  neg_gain_norm,  shift_tot;
  Word16       high_pulse_density_flag;
```

```
PvqEntry_fx entry;
```

```
/* Crear vector absoluto y calcular suma de vector abs */
L_xsum   =  L_deposit_h(0);
max_xabs =  -1;                move16();


FOR( i =0; i < dim; i++)
{
    xabs[i]      = abs_s(x[i]);                    move16();              /* Q12 */
    max_xabs    = s_max(max_xabs,  xabs[i] );              /* para escalado de correlación de búsqueda eficiente */
    L_xsum       = L_mac0(L_xsum, 1, xabs[i] );                    /* permanecer en Q12 */
    y[i]         =0;                          move16();           /* inic. */
}
```

```
test();
IF( L_xsum ==0    ||   neg_gain ==0 )
{  /*  entrada cero o caso de ganancia cero */
  /*  poner una suma de “medios” pulsos primero y último o todos los pulsos en pos. 0 */
  pulse_tot  = pulses;         move16();
  dim_m1    = sub(dim,1);
  y[dim_m1]  = 0;              move16();
  y[0]        = shr(pulses, 1); move16();
  y[dim_m1]  = add(y[dim_m1], sub(pulses, y[0]));  move16();
  L_yy         = L_mult(y[0],y[0]);  /* L_yy necesario para normalización */
  if(dim_m1  !=0) {
    L_yy = L_mac(L_yy, y[dim_m1],y[dim_m1]);   /*( basop único) */
  }
}
```

```
ELSE

{ /* búsqueda habitual */
  /* proj_fac_flt = (pulses - 1)/xsum_flt; */  /* normalizar a norma L1 = pirámide k-1 absoluta */
  num = sub(pulses, 1);
  high_pulse_density_flag   =0; move16();
  if ( sub(pulses,  shr(dim,  1)) >0)
  {
    high_pulse_density_flag = 1; move16();
  }


  test();
  IF( (num >0) && (high_pulse_density_flag  !=0))
  {
    shift_den = norm_l(L_xsum);                          /* entrada x_sum Q12                */
```

```
  den        = extract_h(L_shl(L_xsum,  shift_den)); /* ahora en Q12+shift_den                  */

  L_num      = L_deposit_l(num);
  shift_num = sub(norm_l(L_num)     ,1);
  L_num      = L_shl(L_num, shift_num) ;          /* ahora en Q0 +shift_num -1                */
  proj_fac   = div_l(L_num,   den);               /* L_num siempre tiene que ser menor que den<<16 */


  shift_delta=sub(shift_num,shift_den);
  L_proj_fac = L_shl(L_deposit_l(proj_fac),  sub(9, shift_delta));  /* llevar a un Q12 fijo  */
}

  pulse_tot = 0;              move16();
  L_yy = L_deposit_l(0);
  L_xy = L_deposit_l(0);

  /* Encontrar una posición de inicio en una subpirámide inferior, si la densidad de pulsos es mayor que 0,5 */
  IF((num > 0) &&  (high_pulse_density_flag  !=0))
  {
    FOR( i =0; i < dim ; i++) /* máximo 64 */
    {
      /* y_flt[i]   = (short)floor(xabs_flt[i] * proj_fac_flt); */                        /* truncado de pirámide FLOAT  */
      /* y[i]       =  shr(xabs[i]*L_proj_fac);              */                      /* truncado de pirámide BASOP  */
      Mpy_32_ 16_ss(L_proj_fac,xabs[i],&L_tmp,&u16_tmp);                                  /*Q12 *Q12  +1 */
      y[i]        = extract_l(L_shr(  L_tmp, 12+12-16+1 )); move16();  /* el "truncado" de pirámide */
                                                  /* Q12 *Q12 ->  00 */


      pulse_tot   = add(pulse_tot, y[i]);                 /* Q0                                          */
      L_yy        = L_mac(L_yy, y[i], y[i]);              /* Resultado de energía aumentará en 2 por L_mac */
      L_xy        = L_mac(L_xy, xabs[i], y[i]);               /* Corr, Q0*Q12  +1 --> Q13                        */
    }
  }

  /*  Descripción de bucle de búsqueda fina de PVQ              */
  /*  Ejecutar una búsqueda de CorrSq/Energy  completa de tipo ACELP */
  /*  la lógica de búsqueda básica es para probar añadir un pulso unidad a cada posición n en el vector y de
    salida y actualizar los términos de correlación y energía iterativamente.


  Rxy(k,n)   = Rxy(k-1) + 1*abs(x(k,n))      % añadir correlación unidad de veces el objetivo
  Ryy(k,n)   = Ryy(k-1) + 2*y(k-1,n) + 1^2   % contribución de energía unidad añadida
  RxySq(k,n) = Rxy(k,n)*Rxy(k,n)
```

minimizar (-Rxy(k,n)/Sqrt(Ryy(k,n))), sobre n y k
(o maximizar equivalente (RxySq(k,n)/Ryy(k,n)), sobre n y and k)

RxySq(k,n) bestRxySq
---------- > ---------- ? actualizar mejor candidato
Ryy(k,n) bestRyy

bestRyy * RxySq(k,n) > bestRxySq * Ryy(k,n) ? actualizar mejor candidato

Misc.:
1^2 en Ryy(k,n) se puede añadir antes del bucle de dimensión
Ryy(k,n) se puede disminuir previamente de manera opcional en 2,0 para evitar una multiplicación/cambio
Rxy(k,n) se puede escalar dinámicamente para cada bucle de pulso unidad para optimizar RxySq.
CorrSqTerm =(Rxy(k,n)^2, debería tener una precisión máxima en una variable Word16/Word32

Término de energía:Ryy(k, n)
no debería llegar a ser cero, es decir, muy a menudo tiene suficiente precisión para no llegar a ser truncado
es decir, usar preferiblemente una variable de 32 bits si se acumulan en Q1 más de 127 pulsos.

k=127=>log2(127^2) = 13,9774 bits, encaja en Q1 con signo, con Q1 la división previa por 2 trabajos

k=255=>log2(255^2) = 15,9887 bits, es decir escalar 1 bit, encaja en valor de 16 bits de Q1 sin signo

Aspectos de dimensión:
tenue 6 y más alto siempre es de ahorro de energía de 16 bits ya que los pulsos máximos son 96
(para palabras de código cortas de 1+31 bits)
tenue 5, 4, 3, 2 se puede asignar mayor que 127 pulsos (o menor)
tenue 1 no necesita realmente una búsqueda, solamente es relevante el signo.
Actualmente tenemos KMAX=512, o con más precisión para DIM 5 kmax es 238
DIM 4,3,2,1 kmax es 512
También se podría usar una aritmética de DSP sin signo,
pero entonces el caso de “arranque”/ahora un centinela negativo tiene que ser manejado de una forma especial y además UL_mac,UL_mac0,UL_msu,UL_msu0 actualmente no está disponible como basops sin signo de STL

Para hacer el trabajo de código de búsqueda de PVQ eficientemente hacemos:
1. Siempre usar escalado ascendente dinámico basado en la norm_l casi óptimo del término de correlación, analizando la correlación acumulada hasta ahora. (Rxy para k-1 pulsos)
y usar un valor objetivo máximo posible analizado previamente max_xabs.
2. Usar selectivamente una representación de 16 o 32 bits del término de energía, dependiendo de la energía acumulada previamente.
Además si el término de energía Ryy necesita más de 16 bits, conmutamos a una representación de energía de 32 bits
y también una precisión de 32 bits más alta para la correlación Rxy y

```
            aumentar la precisión en los cálculos de multiplicación cruzada de distorsión a casi 64 bits.

  */
  /* Rxy_max=(k,*)= max( L_xy + 1*max_x_abs)
  análisis necesario para normalización L_xy (corr, corrSq) casi óptima                    */
  /* Ryy_max(k,*) = max(L_yy(k-1) + 0,5  + 1*max_amp_y(k-1 ))
  análisis necesario para selección de precisión de energía (L_yy) cuando los pulsos > 127 ,
  ( (127^2)*2 encaja en Word16 con Signo) */

  L_yy=L_shr(L_yy, 1);  /* disminuir por 2 para dominio de bucle de búsqueda  */
  IF (sub(pulses,127)<=0)
  {   /* bucle interior LC, entra aquí siempre para dimensiones 6 y más altas,
       y también algunas veces para dimensiones 1 .. 5  */
     /* ( si la precisión de energía está inactiva, no se necesita  max_amp_y,
       ni actualizar max_amp_y(k-1)  )                    */
    FOR (k=pulse_tot; k<pulses;k++){
      L_yy = L_add(L_yy, 1); /* añadido 0,5 a energía para pulse_tot+1 */
      imax = one_pulse_search(dim, xabs,y,&pulse_tot,&L_xy,&L_yy,0, 0,max_xabs);
    }
  }
  ELSE
  { /* bucles interiores HC o LC+HC */
    max_amp_y = max_val_fx(y, dim); /* esto hace bucles por encima de 5 valores máximo  */
    /* se necesita max_amp_y a partir de y proyectada cuando pulses_sum  excede 127            */

    /* La primera sección con energía de 32 bits inactiva,   max_amp_y se mantiene actualizada sin embargo  */
    FOR( k=pulse_tot; k<128; k++){
      L_yy        = L_add(L_yy, 1);      /* añadido 0,5 */
      imax        = one_pulse_search(dim, xabs,y,&pulse_tot,&L_xy,&L_yy,0,0,max_xabs);
      max_amp_y   = s_max(max_amp_y, y[imax]);
    }

    /* Segunda sección con número de pulsos más alto, precisión de energía de 32 bits seleccionada
       adaptativamente, max_amp_y mantenida actualizada    */
    FOR( k=pulse_tot; k<pulses; k++){
      L_yy        = L_add(L_yy, 1);   /* añadido 0,5  */
      en_margin   = norm_l(L_mac(L_yy,1,  max_amp_y)); /*encontrar "adición" máxima, margen,~2 op.
*/
      en_dn_shift = sub(16, en_margin);  /*calc. cambio a palabra inferior */

      high_prec_active = 1;      move16();
      if( en_dn_shift <=0){  /* usar solamente energía de 32 bits si se necesita realmente */
        high_prec_active =0;  move16();
```

```
        }
        /* activar energía y corr de 32 bit adaptativamente,  max_amp_y mantenida actualizada  */
        imax      = one_pulse_search(dim,xabs,y,&pulse_tot,&L_xy,&L_yy,high_prec_active,
                                en_dn_shift, max_xabs);
        max_amp_y   = s_max(max_amp_y, y[imax]);
      }
    }
  L_yy = L_shl(L_yy, 1); /*  yy= yy*2,0  */  /* compensar cambio descendente de energía de análisis de bucle de
                                búsqueda en 1,  para hacer energía correcta para cálculo de ganancia unidad/inversa */
}
```

```
/* Aplicar escalado de normalización (L2) de energía unidad, siempre que se necesite comprobar al menos un pulso
   de manera no dividida por cero */
  L_isqrt            = L_deposit_l(0);
  IF( neg_gain != 0 ){
    L_isqrt = lsqrt(L_shr(L_yy, 1));            /* Nota: un factor de ganancia único como en el ajuste no se calcula */
  }

  shift_num        = norm_s(pulse_tot);                    /* cuenta para amplitud de pulso máxima posible en y,
                                                            se puede usar incluso cuando max_amp_y no esté disponib. */
  shift_den        = norm_s(neg_gain);                     /* cuenta para cambio de reducción de escala de ganancia */
  neg_gain_norm = shl(neg_gain, shift_den);            /* hasta 10 dB de pérdida sin esta norma  */
  shift_tot          = sub(add(shift_num, shift_den), 15);
  /* DSP OPT : para ahorrar complejidad media se puede hacer un bucle especial para el caso de ganancia común
==1,0 */
  /* DSP OPT PROM : reutilizar un bucle de normalización de decodificador común                                  */

  L_isqrt = L_negate(L_isqrt);
  FOR( i =0; i < dim; i++)
  {
    tmp = shl(y[i],shift_num);  /* abs(y[i]) cambiado ascendentemente usado en escalado */
    if( x[i] <0)
    {
        tmp = negate(tmp);                                /* aplicar signo */
    }
    if( y[i] !=0 )
    {
      y[i]     = shr(tmp, shift_num);   move16();  /* actualiza signo de y[i},~intervalo -/+ 512),mover disposición */
    Mpy_32_16_ss( L_isqrt, tmp, &L_tmp, &u16_tmp);            /* Q31*Q(0+x)  +1            */
    Mpy_32_ 16_ss( L_tmp, neg_gain_norm, &L_tmp, &u16_tmp);   /* Q31*Q(0+x) *Q15 +1          */
    L_tmp   = L_shr(L_tmp, shift_tot);                    /* Q31+x   */
    xq[i]  = round_fx(L_tmp);          move16();            /*Q15, mover disposición     */
    L_xq[i] = L_tmp;                                  /* Q31 actualmente sin uso  */
```

```
}
/* indexar el vector de PVQ encontrado en palabras de código cortas */
entry = mpvq_encode_vec_fx(y, dim, pulses);
/* enviar la(s) palabra(s) de código corta(s) al codificador de intervalo */
rc_enc_bits_fx(st_fx, UL_deposit_l(entry.lead_sign_ind) , 1);  /* 0 o 1 */
IF( sub( dim, 1) !=0)
{
rc_enc_uniform_fx(st_fx, entry.index, entry.size);
}
return;
```

El código anterior debería ser fácil de leer para todas las personas expertas en la técnica y no debería ser explicado en más detalle. No obstante, para las personas no expertas se menciona que el operador relacional "= =" es un operador que en un ejemplo de "A= =B" devuelve un valor lógico fijado a un 1 lógico (verdadero) cuando los valores A y B son iguales; y de otro modo devuelve un 0 lógico (falso). L_mac es un multiplicar-acumular dentro del significado de que L_mac (L_v3, v1, v2) = L_v3+v1*v2.

Apéndice 2: Resultados de simulación puestos en tabla

Antecedentes de la simulación

Se han simulado realizaciones de la descripción en la presente memoria. Para todas las simulaciones de búsqueda de forma de PVQ hechas, la tasa de bit usada fue 64.000 bps y el códec se operó en modo MDCT, con tamaños de subbanda de coeficiente MDCT inicial de [8, 12, 16, 24, 32] coeficientes. Estas bandas se pueden dividir muy bien en secciones de banda menores, cada una representada por un subvector, por un algoritmo de división de banda de PVQ. Por ejemplo, una banda de tamaño 8 se puede dividir en subsecciones más pequeñas, por ejemplo, "4, 4" o "3, 3, 2", si se asignan suficientes bits. Típicamente, cada banda se divide de tal forma que se pueden usar un máximo de 32 bits para codificación de forma de cada subvector final.

En esta implementación de indexación de PVQ una banda de tamaño 8 puede tener hasta 36 pulsos unidad, una subsección de tamaño 7 puede tener hasta 53 pulsos unidad, una sección de tamaño 6 puede tener hasta 95 pulsos unidad, una sección de tamaño 5 puede tener hasta 238 pulsos unidad, una sección de tamaño 4, 3, 2, 1 puede tener hasta 512 pulsos unidad. Como las secciones más cortas con un alto número de pulsos se crean dinámicamente dividiendo la banda, son menos frecuentes que los tamaños de subvectores más largos. Las cifras WMOPS en las Tablas de Resultados de más adelante incluyen: (Búsqueda previa de PVQ, búsqueda fina de PVQ, normalización de PVQ e indexación de PVQ). Las cifras "idéntico %" en las Tablas de Resultados de más adelante, es el número de vectores idénticos encontrados en el Algoritmo de búsqueda de forma de precisión limitada evaluada, comparado con una búsqueda de forma de PVQ de punto flotante no restringido.

Tablas de resultados

Tabla 1 Resultados para K final <= 127

Pulsos <= 127, **Algoritmo** En{energy-bits} x CorrSq{corrSq-bits}	**SNR Min (dB)**	**SegSNR (dB)**	**vectores idénticos %**	**WMOPS de Caso Peor**	**WMOPS Media**	**Observación**
"En16xCorrSq16/"En32x CorrSq32' mezclado, pre_analyze max(x_abs)	4,771	188,803	99,3	6,843	5,496	16x16 siempre usado, WC (peor caso) en 16x16
"***En16xCorrSq16"*** bloqueado pre_analyze max(x_abs)	4,771	188,803	99,3	6,843		Ningún cambio ya que la energía nunca excede 16 bits

Pulsos <= 127, **Algoritmo** En{energy-bits} x CorrSq{corrSq-bits}	**SNR Min (dB)**	**SegSNR (dB)**	**vectores idénticos %**	**WMOPS de Caso Peor**	**WMOPS Media**	**Observación**
"En16 x CorrsSq16" que usa un método de escalado de correlación de la técnica conocida "OPUS", usando un número de pulsos unidad acumulado.	-6,021	180,556	94,6	6,826	5,476	El algoritmo es el peor bit (minSNR menor menos impactos idénticos) en una complejidad muy similar
"En32 x CorrSq16" bloqueado***,*** analizar previamente max(x_abs)	4,771	188,803	99,3	8,970	6,961	Innecesario usar En32 para pulsos <=127, ya que la energía nunca excede dinámica de 16 bits
"En16 x CorrSq32" bloqueado, analizar previamente entrada max(x_abs)	190,0	190,0	100	9,386	7,248	2,5 WMOPS extra requeridas para los últimos 0,7% impactos
"En32 x CorrSq32" bloqueado, analizar previamente max(x_abs)	190,0	190,0	100	10,474	7,999	0,9 WMOPS innecesarias aumentan comparado con ***"En16xCorrSq32"*** Bloqueado

Tabla 2 Resultados para K > 127

Pulsos > 127, **Algoritmo** En{energy-bits} x CorrSq{corrSq-bits}	**minSNR (dB)**	**segSNR (dB)**	**vectores idénticos %**	**WMOPS de Caso Peor**	**WMOPS Media**	**Observación**
AccEn mezclado controlado***"En16xCorrSq16'/"En32x CorrSq32',*** pre_analyze entrada, precisión controlada de energía acumulada	32,686	160,316	80,4%	6,843 (WC aún desde secciones 16x16)	5,496	Una solución bastante buena WC aún es para 16x16, WC no se aumenta
AccEn mezclado controlado***"En16xCorrSq16'/ "En16x CorrSq32'*** pre_analyze entrada, precisión controlada de energía acumulada	32,686	130,258	59,3%	n/a	n/a	La información de energía se trunca ocasionalmente causando SNR baja
AccEn mezclado controlado***"En16 x CorrSq16'/"En32x CorrSq16'*** pre_analyze entrada, precisión controlada de energía acumulada	32,686	117,634	50,6%	n/a	n/a	La información de correlación tiene baja precisión, causando baja SNR

Pulsos > 127, Algoritmo En{energy-bits} x CorrSq{corrSq-bits}	**minSNR (dB)**	**segSNR (dB)**	**vectores idénticos %**	**WMOPS de Caso Peor**	**WMOPS Media**	**Observación**
"En16 x CorrSq16' bloqueado, pre_analyze entrada,	32,686	113,629	47,8%	n/a	n/a	La información de energía truncada ocasionalmente y la información de correlación tiene baja precisión, causando SNR baja
"En32 x CorrSq16' bloqueado, pre_analyze entrada	32,686	117,634	50,6%	n/a	n/a	La información de correlación tiene baja precisión, causando SNR baja
"En16 x CorrSq32' bloqueado, pre_analyze entrada	40,994	159,714	78,8%	n/a	n/a	La información de energía se trunca ocasionalmente , causando SNR baja
"En32 x CorrSq32' bloqueado, pre_analyze entrada	49,697	189,773	99,8%	**7,1**	**5,7**	WC ahora en sección 32x32, WC de complejidad más alta

REIVINDICACIONES

1. Un método para búsqueda de forma de Cuantificador de Vector en Pirámide, PVQ, realizada por un codificador, el PVQ que toma un vector objetivo x como entrada y que deriva un vector y añadiendo iterativamente pulsos unidad en un bucle de búsqueda de dimensión interior, el método que se caracteriza por:

antes de entrar en un siguiente bucle de búsqueda de dimensión interior para adición de pulso unidad:

- determinar (102, 204), en base a una amplitud de pulso máxima, $maxamp_y$, de un vector actual y, si se necesita más de una longitud de palabra de bit actual para representar, de una manera sin pérdidas, una variable, $enloop_y$, relacionada con una energía acumulada de y, en el bucle de dimensión interior próximo (103).

2. El método según la reivindicación 1, en donde el método además comprende:

antes de entrar en un siguiente bucle de dimensión interior para adición de pulso unidad:

- determinar (203), en base al valor absoluto máximo, $xabs_{max}$ del vector de entrada, x, un cambio ascendente posible, en una palabra de bit, del valor de correlación dentro del bucle acumulado del siguiente bucle, $corr_{xy}$, entre x y el vector y.

3. El método según la reivindicación 1 o 2, que además comprende:

cuando se necesita más de una longitud de palabra de bit actual (102, 204, 304, 403) para representar $enloop_y$:

- realizar (103, 205, 305, 404) los cálculos de bucle interior usando una longitud de palabra de bit más larga para representar $enloop_y$.

4. El método según cualquiera de las reivindicaciones 1-3, que además comprende:

cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$:

- realizar (305) los cálculos de bucle interior usando una longitud de palabra de bit más larga para representar un valor de correlación dentro del bucle acumulado cuadrado, $corr_{xy}^2$, entre x y el vector y, en el bucle.

5. El método según cualquiera de las reivindicaciones precedentes, que además comprende, cuando no se necesita más de una longitud de palabra de bit actual para representar $enloop_y$:

- realizar (405) los cálculos de bucle interior empleando un primer bucle de adición de pulso unidad usando una primera longitud de palabra de bit para representar $enloop_y$ y:

cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$:

- realizar (404) los cálculos de bucle interior empleando un segundo bucle de adición de pulso unidad usando una longitud de palabra de bit más larga para representar $enloop_y$ que el primer bucle de adición de pulso unidad.

6. El método según cualquiera de las reivindicaciones precedentes, que además comprende, cuando no se necesita más de una longitud de palabra de bit actual para representar $enloop_y$:

- realizar (405) los cálculos de bucle interior empleando un primer bucle de adición de pulso unidad que tiene una cierta precisión y:

cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$:

- realizar (404) los cálculos de bucle interior empleando un segundo bucle de adición de pulso unidad que tiene una precisión mayor que el primer bucle de adición de pulso unidad.

7. El método según cualquiera de las reivindicaciones precedentes, en donde la determinación, basada en $maxamp_y$, de si se necesita más de una longitud de palabra de bit actual para representar $enloop_y$ comprende determinar las características del caso cuando, en el bucle de búsqueda interior próximo, el pulso unidad se añade a la posición en y que está asociada con $maxamp_y$.

8. El método según cualquiera de las reivindicaciones precedentes, que además comprende:

en el bucle de búsqueda de dimensión interior para adición de pulso unidad:

- determinar una posición, n_{best}, en y para adición de un pulso unidad evaluando una multiplicación cruzada, para cada posición n en y, de un valor de correlación y energía para la n actual; y un valor de correlación cuadrada, BestCorrSq y uno de energía, bestEn, guardados a partir de valores previos de n; como:

$$corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

donde

$$\left.\begin{array}{r} n_{best} = n \\ bestEn = enloop_y \\ BestCorrSq = corr_{xy}^2 \end{array}\right\} \text{cuando } corr_{xy}^2 * bestEn > BestCorrSq * enloop_y$$

9. El método según cualquiera de las reivindicaciones precedentes, que además comprende,

- hacer un seguimiento de $maxamp_y$ cuando un valor final de K, asociado con el vector objetivo x, excede un valor umbral.

10. Un codificador configurado para búsqueda de forma de Cuantificación de Vector en Pirámide, PVQ; la PVQ que toma un vector objetivo x como entrada y que deriva un vector y añadiendo iterativamente pulsos unidad en un bucle de búsqueda de dimensión interior, el codificador que se caracteriza por estar configurado para:

antes de entrar en un siguiente bucle de búsqueda de dimensión interior para adición de pulso unidad:

- determinar, en base a una amplitud de pulso máxima, $maxamp_y$, de un vector actual y, si se necesita más de una longitud de palabra de bit actual para representar, de una manera sin pérdidas, una variable, $enloop_y$, relacionada con una energía acumulada de y, en el bucle de dimensión interior próximo.

11. El codificador según la reivindicación 10, que además se configura para:

antes de entrar en un siguiente bucle de dimensión interior para adición de pulso unidad:

- determinar, en base al valor absoluto máximo, $xabs_{max}$ del vector de entrada, x, un cambio ascendente posible, en una palabra de bit, del valor de correlación dentro del bucle acumulado del siguiente bucle, $corr_{xy}$, entre x y el vector y.

12. El codificador según la reivindicación 10 u 11, que además se configura para:

- realizar los cálculos de bucle interior usando una longitud de palabra de bit más larga para representar $enloop_y$, cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$.

13. El codificador según cualquiera de las reivindicaciones 9-12, que se configura para:

- realizar los cálculos de bucle interior empleando un primer bucle de adición de pulso unidad usando una primera longitud de palabra de bit cuando no se necesita más de una longitud de palabra de bit actual para representar $enloop_y$ y:

- realizar los cálculos de bucle interior empleando un segundo bucle de adición de pulso unidad usando una longitud de palabra de bit más larga que el primer bucle de adición de pulso unidad cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$.

14. El codificador según cualquiera de las reivindicaciones 9-13, que además se configura para:

- realizar los cálculos de bucle interior empleando un primer bucle de adición de pulso unidad, que tiene una cierta precisión, cuando no se necesita más de una longitud de palabra de bit actual para representar $enloop_y$; y para

- realizar los cálculos de bucle interior empleando un segundo bucle de adición de pulso unidad, que tiene una precisión mayor que el primer bucle de adición de pulso unidad, cuando se necesita más de una longitud de palabra de bit actual para representar $enloop_y$.

15. El codificador según cualquiera de las reivindicaciones 9-14, en donde la determinación, basada en $maxamp_y$, de si se necesita más de una longitud de palabra de bit actual para representar $enloop_y$ se configura para comprender determinar las características del caso cuando, en el bucle de búsqueda interior próximo, el pulso unidad se añade a la posición en y que está asociada con $maxamp_y$.

16. El codificador según cualquiera de las reivindicaciones 9-15, que además se configura para:

en el bucle de búsqueda de dimensión interior para adición de pulso unidad:

- determinar una posición, n_{best}, en y para adición de un pulso unidad evaluando una multiplicación cruzada, para cada posición n en y, de un valor de correlación y

energía para la n actual; y un valor de correlación, BestCorrSq y

de energía, bestEn, guardados a partir de valores previos de n; como:

$$\mathrm{corr}_{xy}^{2} * \mathrm{bestEn} > \mathrm{BestCorrSq} * \mathrm{enloop}_{y}$$

donde

$$\left.\begin{array}{r} n_{best} = n \\ bestEn = enloop_y \\ BestCorrSq = corr_{xy}^2 \end{array}\right\} \text{cuando } \mathrm{corr}_{xy}^{2} * \mathrm{bestEn} > \mathrm{BestCorrSq} * \mathrm{enloop}_{y}$$

17. El codificador según cualquiera de las reivindicaciones 9-16, que además se configura para hacer el seguimiento de $maxamp_y$ cuando un número de pulsos unidad finales, K, asociado con el vector objetivo x, excede un valor umbral.

18. Un dispositivo de comunicación que comprende el codificador según cualquiera de las reivindicaciones 9-17.

19. Un programa de ordenador, que comprende instrucciones que, cuando se ejecutan en al menos un procesador, hacen al por lo menos un procesador llevar a cabo el método según cualquiera de las reivindicaciones 1-8.

20. Un programa de ordenador según la reivindicación 19, en donde al menos uno del al menos un procesador es un Procesador de Señal Digital, DSP, por ejemplo, un DSP de precisión fija.

21. Un portador que contiene el programa de ordenador de la reivindicación 19 o 20, en donde el portador es uno de una señal electrónica, señal óptica, señal radio o medio de almacenamiento legible por ordenador.

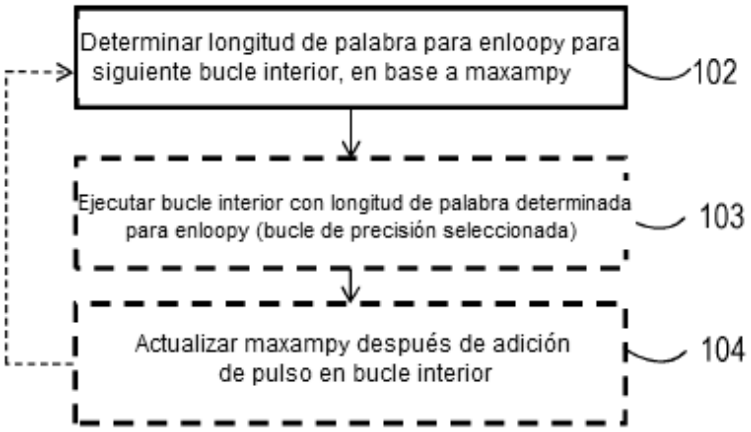

Figura 1

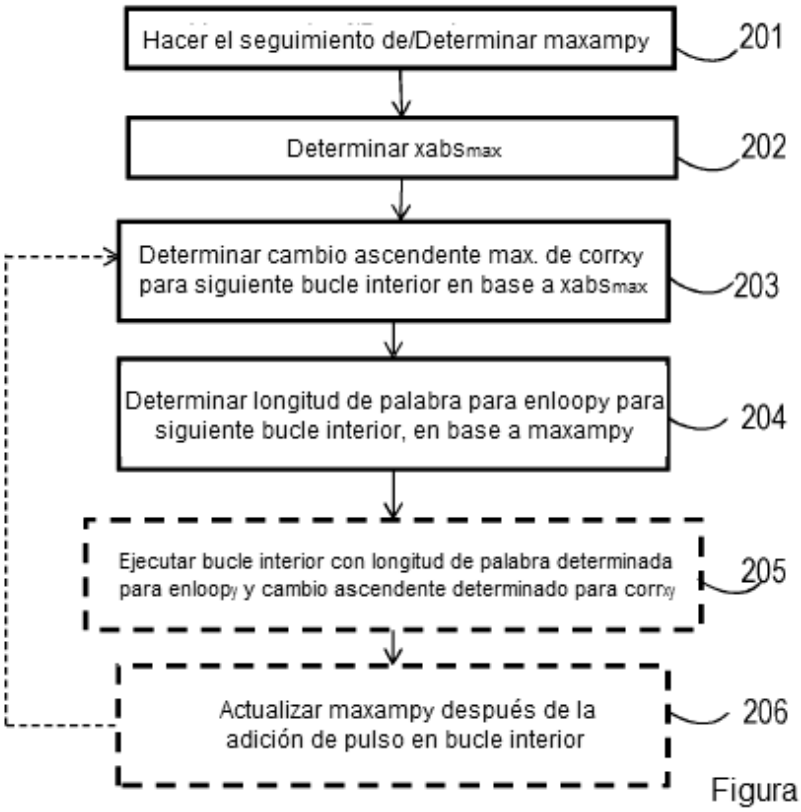

Figura 2

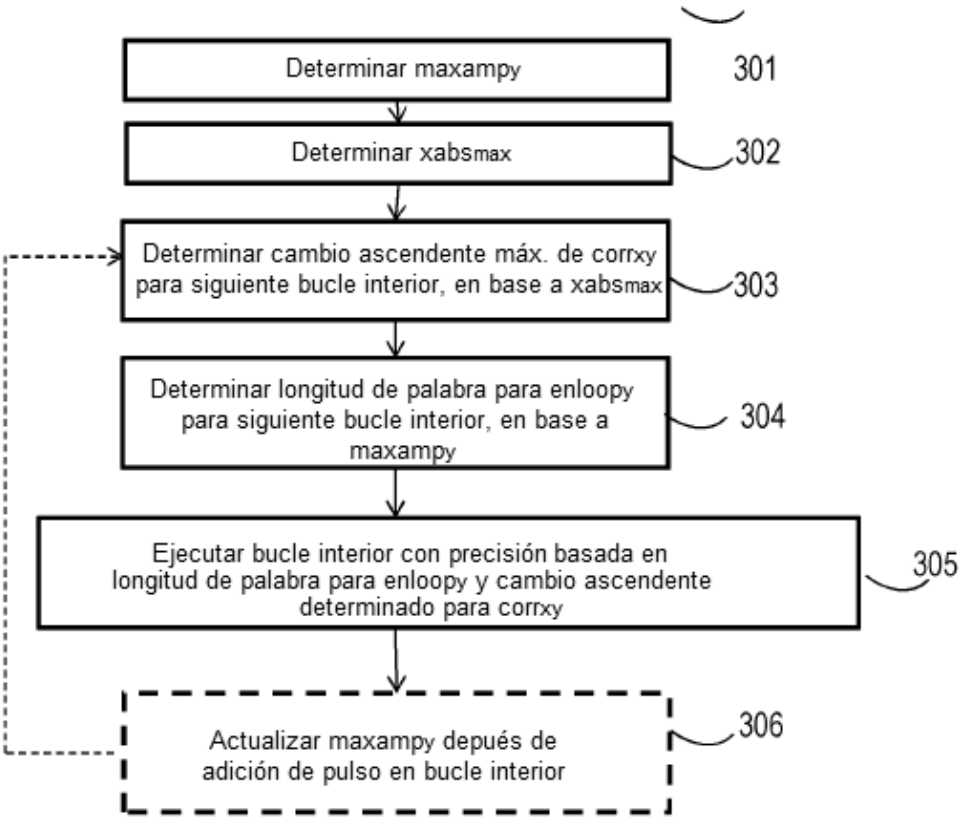

Figura 3

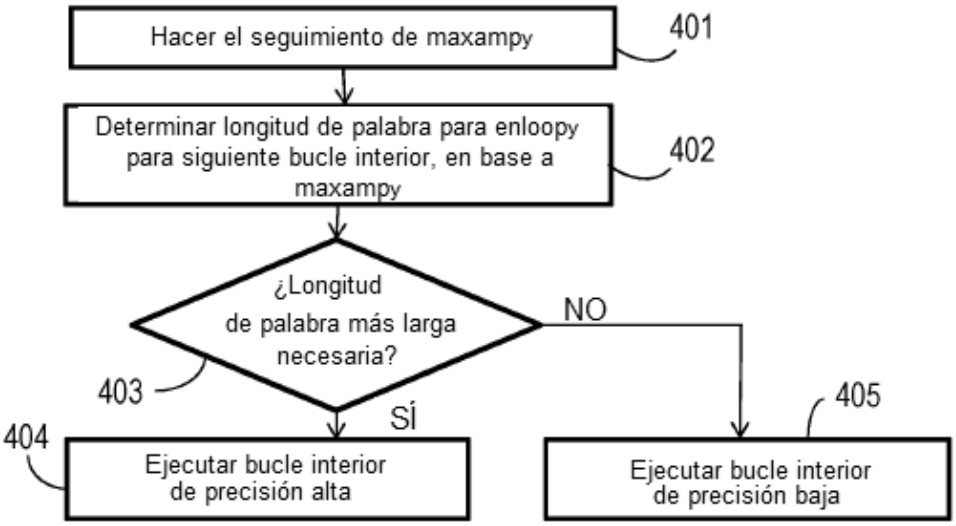

Figura 4

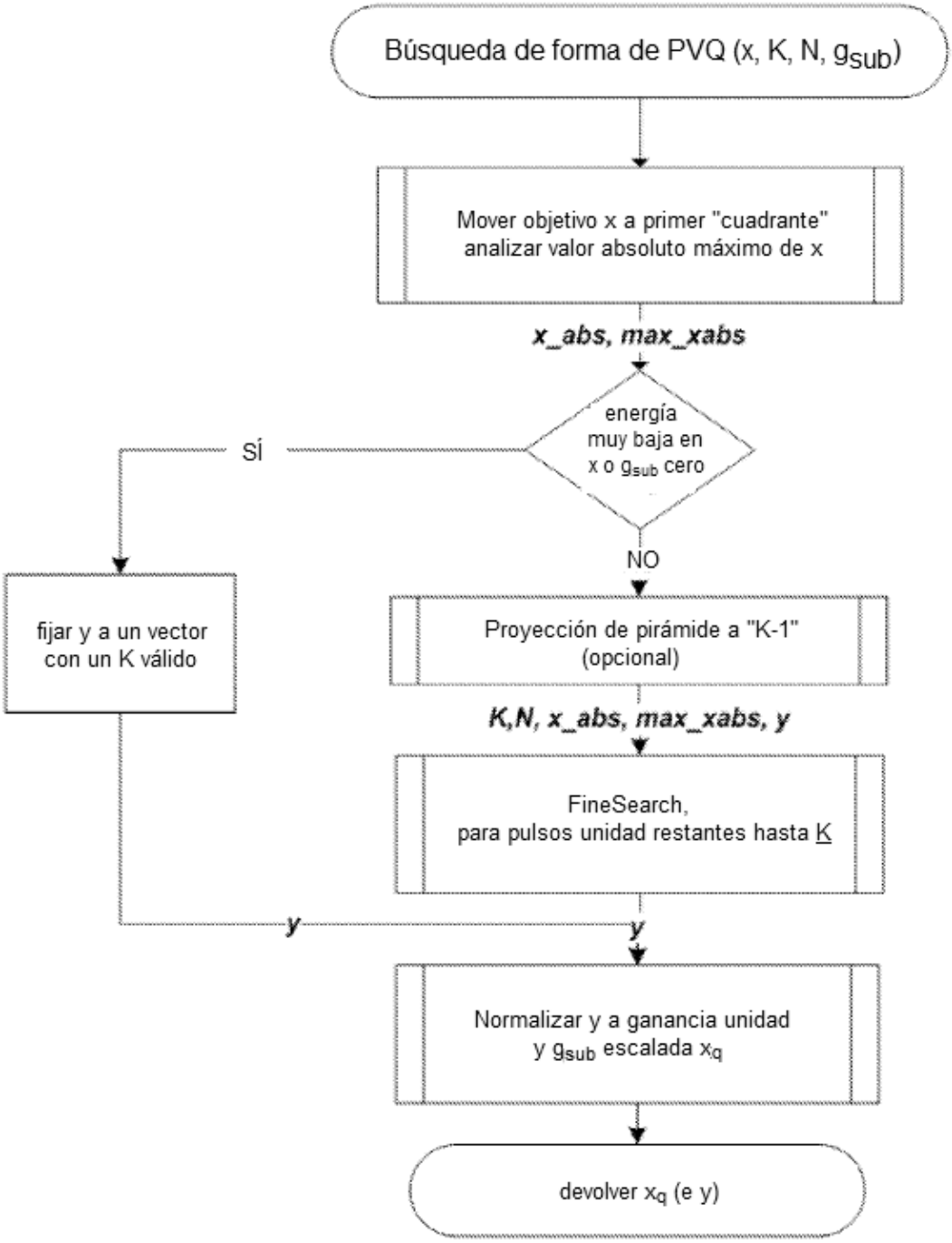
Búsqueda de forma de PVQ (x, K, N, g_{sub})
Mover objetivo x a primer "cuadrante"
analizar valor absoluto máximo de x
x_abs, max_xabs
energía
muy baja en
x o g_{sub} cero
SÍ
NO
fijar y a un vector
con un K válido
Proyección de pirámide a "K-1"
(opcional)
K,N, x_abs, max_xabs, y
FineSearch,
para pulsos unidad restantes hasta K
y
y
Normalizar y a ganancia unidad
y g_{sub} escalada x_q
devolver x_q (e y)

Figura 5

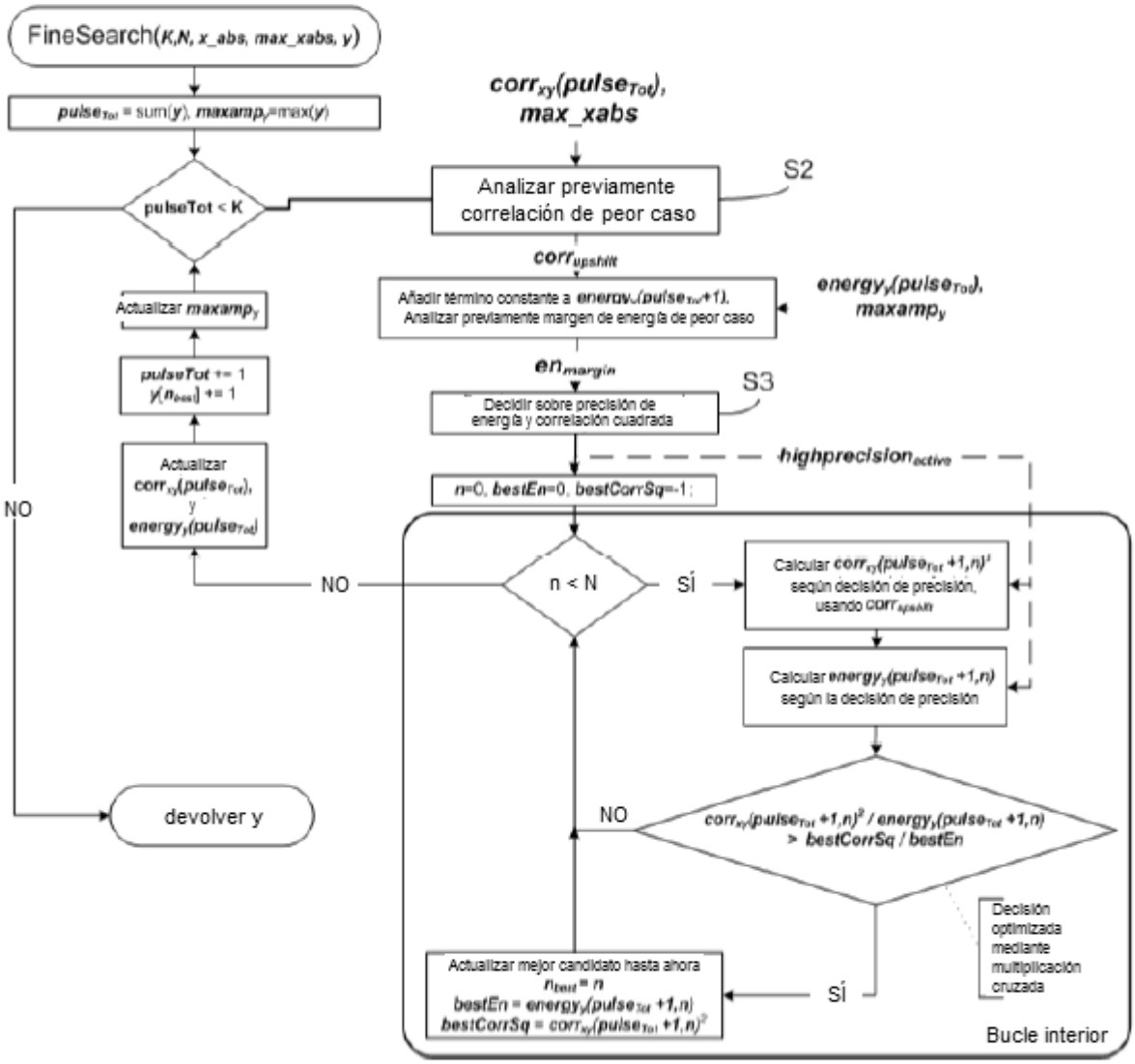
FineSearch(K,N, x_abs, max_xabs, y)
pulseTot = sum(y), maxamp_y=max(y)
pulseTot < K
NO
devolver y
Actualizar maxamp_y
pulseTot += 1
y(n_best) += 1
Actualizar corr_xy(pulse_Tot), y energy_y(pulse_Tot)
NO
corr_xy(pulse_Tot), max_xabs
Analizar previamente correlación de peor caso
S2
corr_upshift
Añadir término constante a energy_y(pulse_Tot+1). Analizar previamente margen de energía de peor caso
energy_y(pulse_Tot), maxamp_y
en_margin
Decidir sobre precisión de energía y correlación cuadrada
S3
highprecision_active
n=0, bestEn=0, bestCorrSq=-1;
n < N
SÍ
Calcular corr_xy(pulse_Tot +1,n)² según decisión de precisión, usando corr_upshift
Calcular energy_y(pulse_Tot +1,n) según la decisión de precisión
NO
corr_xy(pulse_Tot +1,n)² / energy_y(pulse_Tot +1,n) > bestCorrSq / bestEn
Decisión optimizada mediante multiplicación cruzada
SÍ
Actualizar mejor candidato hasta ahora
n_best = n
bestEn = energy_y(pulse_Tot +1,n)
bestCorrSq = corr_xy(pulse_Tot +1,n)²
Bucle interior

Figura 6

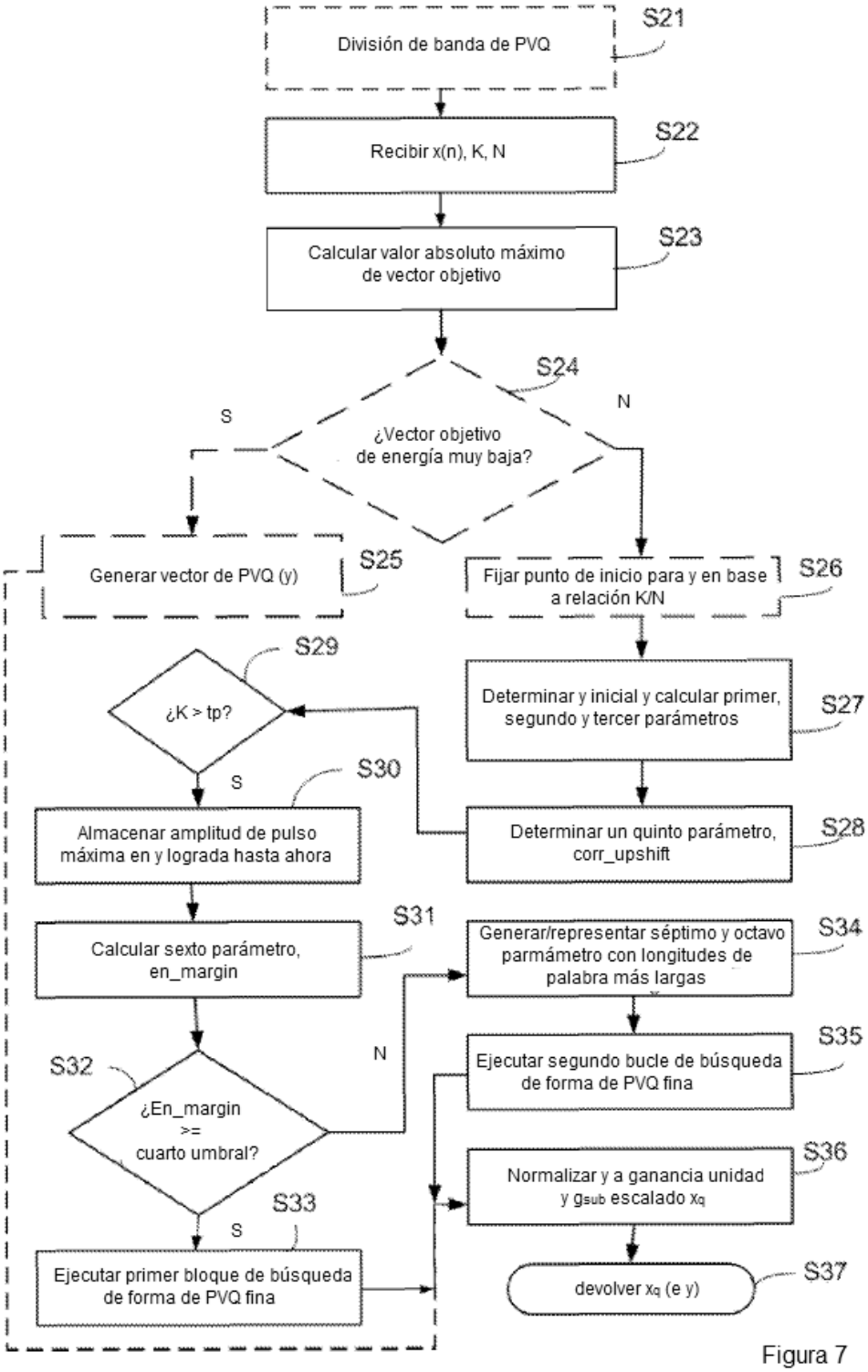
División de banda de PVQ
S21
Recibir x(n), K, N
S22
Calcular valor absoluto máximo de vector objetivo
S23
S24
¿Vector objetivo de energía muy baja?
S
N
Generar vector de PVQ (y)
S25
Fijar punto de inicio para y en base a relación K/N
S26
S29
¿K > tp?
Determinar y inicial y calcular primer, segundo y tercer parámetros
S27
S
S30
Almacenar amplitud de pulso máxima en y lograda hasta ahora
Determinar un quinto parámetro, corr_upshift
S28
S31
Calcular sexto parámetro, en_margin
Generar/representar séptimo y octavo parmámetro con longitudes de palabra más largas
S34
S35
S32
N
¿En_margin >= cuarto umbral?
Ejecutar segundo bucle de búsqueda de forma de PVQ fina
S36
Normalizar y a ganancia unidad y gsub escalado xq
S33
S
Ejecutar primer bloque de búsqueda de forma de PVQ fina
devolver xq (e y)
S37

Figura 7

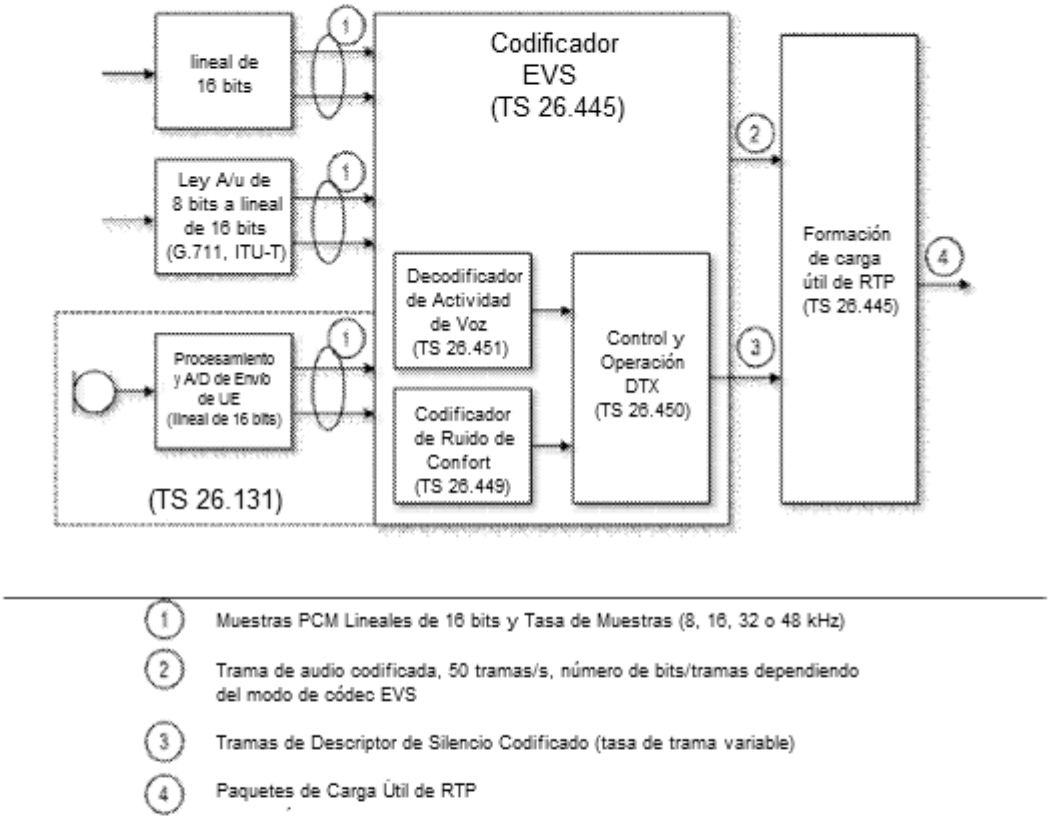
lineal de
16 bits
Ley A/u de
8 bits a lineal
de 16 bits
(G.711, ITU-T)
Procesamiento
y A/D de Envío
de UE
(lineal de 16 bits)
(TS 26.131)
Codificador
EVS
(TS 26.445)
Decodificador
de Actividad
de Voz
(TS 26.451)
Codificador
de Ruido de
Confort
(TS 26.449)
Control y
Operación
DTX
(TS 26.450)
Formación
de carga
útil de RTP
(TS 26.445)
1
2
3
4
1 Muestras PCM Lineales de 16 bits y Tasa de Muestras (8, 16, 32 o 48 kHz)
2 Trama de audio codificada, 50 tramas/s, número de bits/tramas dependiendo del modo de códec EVS
3 Tramas de Descriptor de Silencio Codificado (tasa de trama variable)
4 Paquetes de Carga Útil de RTP

Figura 8

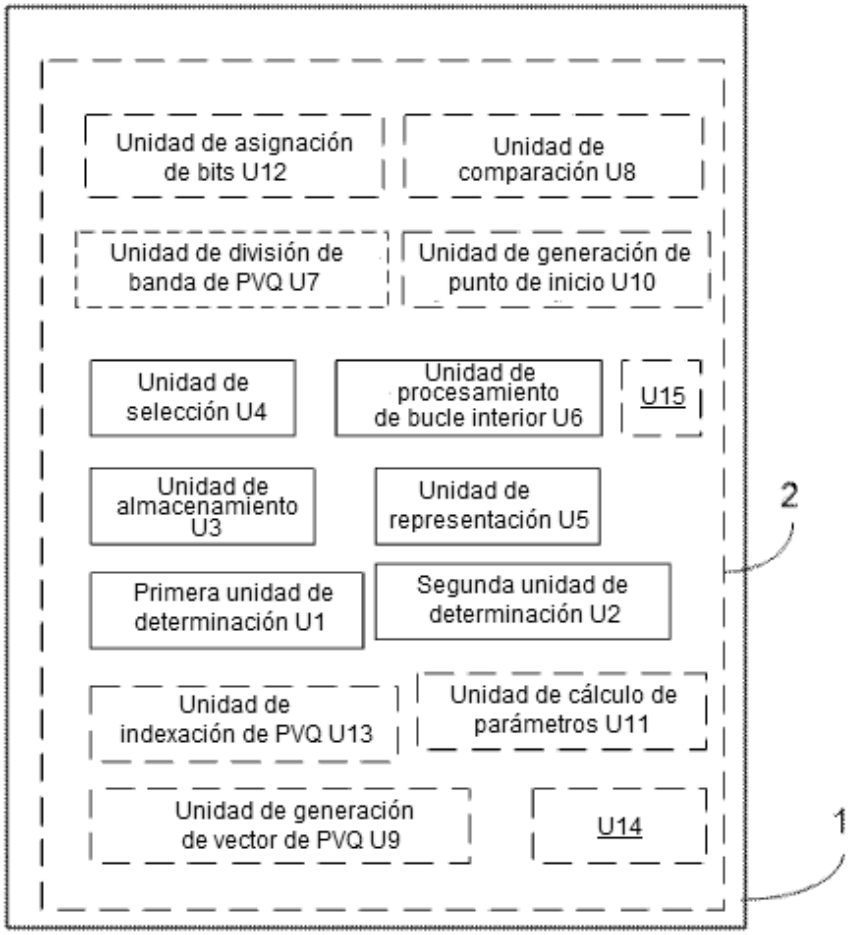
Unidad de asignación de bits U12
Unidad de comparación U8
Unidad de división de banda de PVQ U7
Unidad de generación de punto de inicio U10
Unidad de selección U4
Unidad de procesamiento de bucle interior U6
U15
Unidad de almacenamiento U3
Unidad de representación U5
2
Primera unidad de determinación U1
Segunda unidad de determinación U2
Unidad de indexación de PVQ U13
Unidad de cálculo de parámetros U11
Unidad de generación de vector de PVQ U9
U14
1

Figura 9

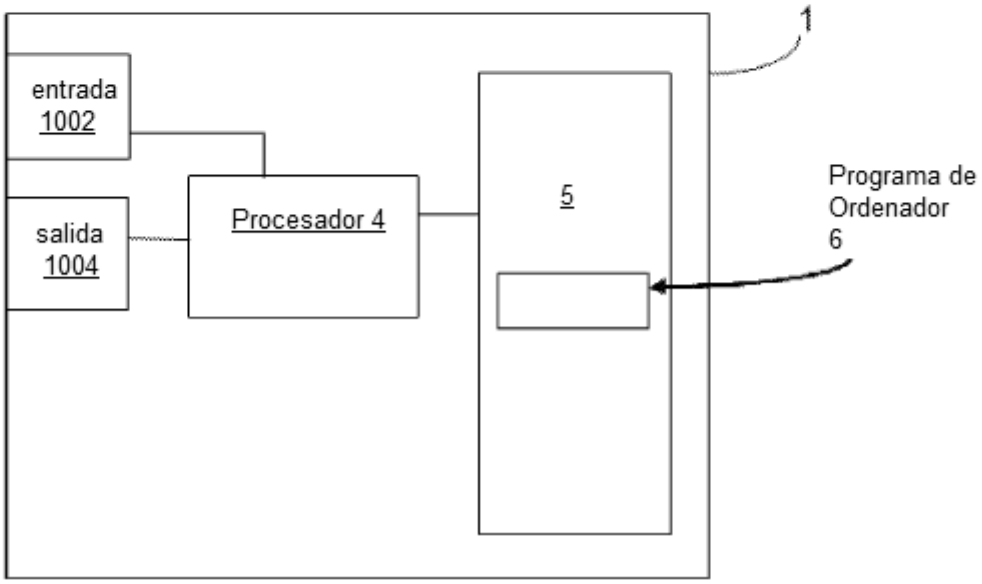
1
entrada
1002
salida
1004
Procesador 4
5
Programa de
Ordenador
6

Figura 10

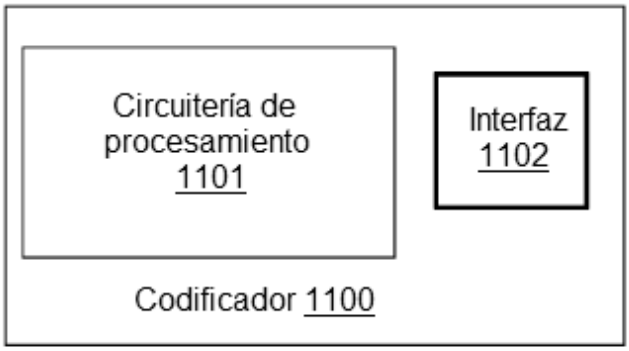

Figura 11a

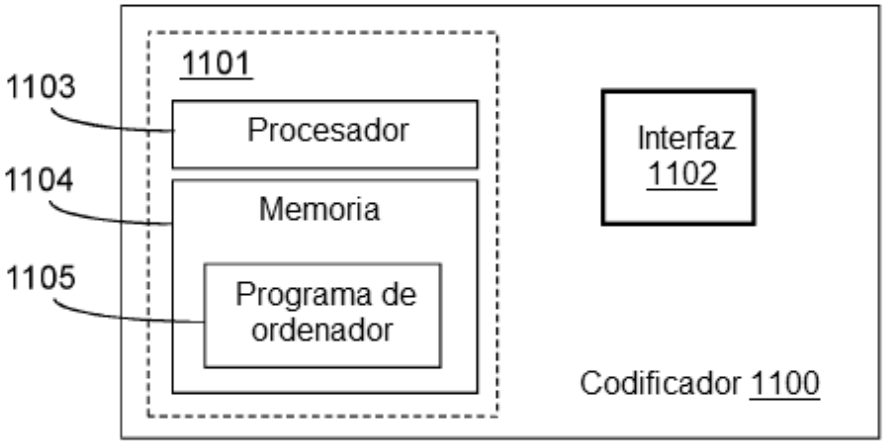

Figura 11b

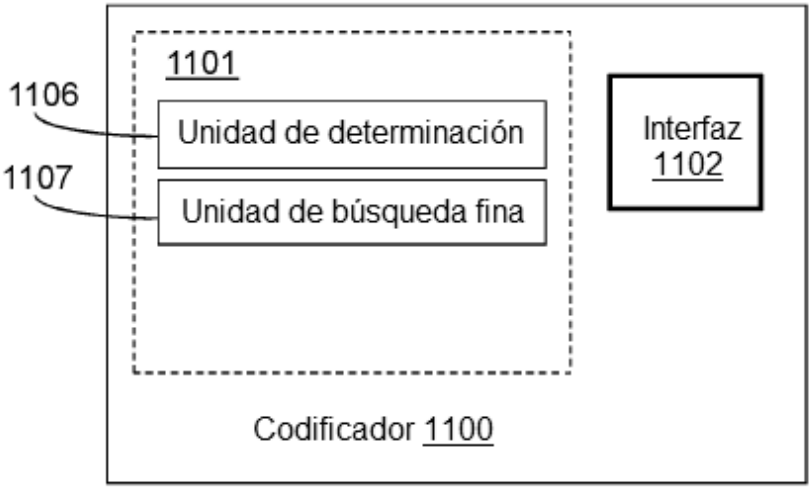

Figura 11c

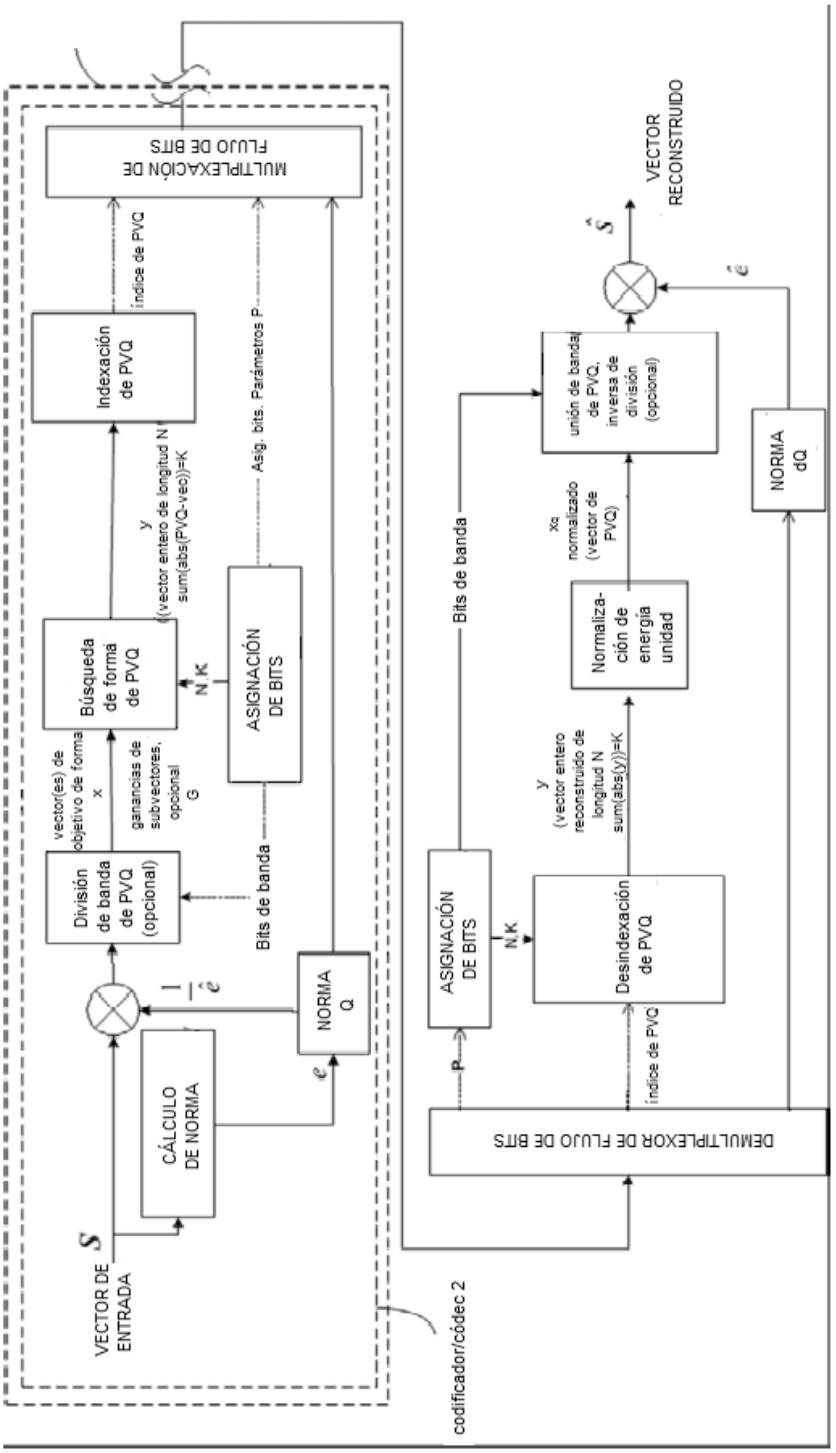
VECTOR DE ENTRADA
CÁLCULO DE NORMA
División de banda de PVQ (opcional)
vector(es) de objetivo de forma x
ganancias de subvectores, opcional G
Búsqueda de forma de PVQ
y (vector entero de longitud N sum(abs(PVQ-vec))=K
Indexación de PVQ
índice de PVQ
MULTIPLEXACIÓN DE FLUJO DE BITS
NORMA Q
Bits de banda
ASIGNACIÓN DE BITS
N,K
Asig. bits. Parámetros P
codificador/códec 2
DEMULTIPLEXOR DE FLUJO DE BITS
P
ASIGNACIÓN DE BITS
N,K
Bits de banda
Desindexación de PVQ
índice de PVQ
y (vector entero reconstruido de longitud N sum(abs(y))=K
Normalización de energía unidad
xq normalizado (vector de PVQ)
unión de banda de PVQ, inversa de división (opcional)
NORMA dQ
VECTOR RECONSTRUIDO

Figura 12